Generation and Distribution of GHZ States in Quantum Networks

Mohammad Ghaderibaneh, Himanshu Gupta, C.R. Ramakrishnan
Department of Computer Science, Stony Brook University, Stony Brook, NY 11790, USA

***Abstract*—In quantum networks, multipartite entangled states distributed over the network are useful in implementing and supporting many quantum network applications for communications, sensing, and computing. The focus of our work is to develop techniques to efficiently generate distributed Greenberger-Horne-Zeilinger (GHZ) states, a special class of multipartite entanglement states. Prior works on generating GHZ states have focused on the objective of minimizing the number of maximally entangled (bipartite) pairs (EPs), while ignoring the stochastic nature of quantum processes and assuming uniform network links for generating EPs. In contrast, in our work, we take into consideration the stochastic nature of quantum networks, and focus on maximizing the expected generated rate of the GHZ states under given fidelity constraints. In this context, we develop two efficient generation schemes, viz., `Fusion-Retain-Only` and `General Fusion`, comprised of optimal or near-optimal sub-steps. Both schemes, at a high-level, first determine a way to "connect" the nodes over which the GHZ state is to distributed, and then, determine a sequence of fusion operations on the EPs created over the connections. Using extensive simulations over a quantum network simulator (NetSquid), we demonstrate the effectiveness of our developed techniques and show that our schemes outperform prior work as well as a simple centralized approach by up to orders of magnitude while generating GHZ states of tolerable fidelity.**

I. Introduction

Fundamental advances in physical sciences and engineering have led to the realization of working quantum computers (QCs) [1]. However, there are significant limitations to the capacity of individual QCs [2]. Quantum networks (QNs) enable the construction of large-scale and robust quantum computing platforms by connecting smaller QCs [3]. QNs also enable various important applications [4]–[14], but to implement and support these applications, we need to create and distribute entangled states efficiently. Recent works [15], [16] have considered on generating and distributing *Bell pairs*, i.e., pairwise maximally entangled states (EPs). However, there are several applications (see below) that also make use of *multipartite* entangled states. Thus, in this work, we focus on the generation and distribution of Greenberger-Horne-Zeilinger (GHZ) states, an important class of multipartite states.

Motivation for Distributing GHZ States. There are many classes of multipartite entanglements with different characteristics and applications, e.g., W and GHZ states. In particular, GHZ states, a generalization of Bell pairs, form a useful class of multipartite maximally entangled states [17]. They are used in many multiparty applications of quantum information such as error correction [18], quantum secret sharing [19], anonymous transmission [20], quantum metrology [21], conference key agreement [22], clock synchronization [23], and extending the baseline of telescopes [24]. Thus, the design of efficient network protocols to generate and distribute GHZ states is important for the implementation of many useful protocols and applications of quantum networks. GHZ states can also be used to generate arbitrary *graph states* which is another useful class of multipartite states [25], [26]. In particular, distributed graph states, as well as GHZ states, can also be used to extract distributed bipartite entanglements which have many applications such as teleportation, long-distance entanglements, and quantum communications [27].

Prior Work and Our Approach. Recently, there have been some works [25], [28]–[32] that have addressed the problem of efficient generation and distribution of GHZ states in a quantum network. These works [25], [28] have focused on minimizing the *number* of EPs consumed. They implicitly ignore the stochastic nature of the underlying processes, and assume distributed EPs to be available *a priori*. A true count of EPs consumed in the generation of GHZ states should take into consideration the stochastic nature of operations (e.g., fusion) involved, particularly, since they can have a relatively low probability of success. Moreover, some EPs may take significantly longer to generate than others, due to the heterogeneity of the network links (e.g. different lengths). Consequently, the count of EPs alone is too simplistic a measure of overall performance.

Following prior works on the generation and distribution of EPs [15], [33]–[35], we consider the more appropriate optimization objective, viz., maximizing the expected generation rate under given fidelity constraints; this objective explicitly takes into consideration the stochastic nature of all processes, heterogeneity of the network links/elements, and limited network resources. Overall, our proposed schemes use *fusion operations* to progressively generate larger GHZ states from smaller GHZ states or EPs. The fusion operations can be seen as generalizations of Bell-State Measurement (BSM) operations used in quantum repeaters for the generation and distribution of EPs. Our proposed schemes have two high-level steps: (i) Determine the best way to "connect" the nodes over which the desired GHZ state is to be distributed, (ii) Determine the sequence of fusion operations on the EPs generated over the connections, to create the final desired GHZ state. In determining the expected rate, our techniques take into consideration "waiting" time incurred by an intermediate (GHZ) state in waiting for its sibling-operand to become available (possibly, after many failed attempts) before a binary quantum operation (entanglement swapping or fusion) can be performed; this is similar to the prior works on EP genera-

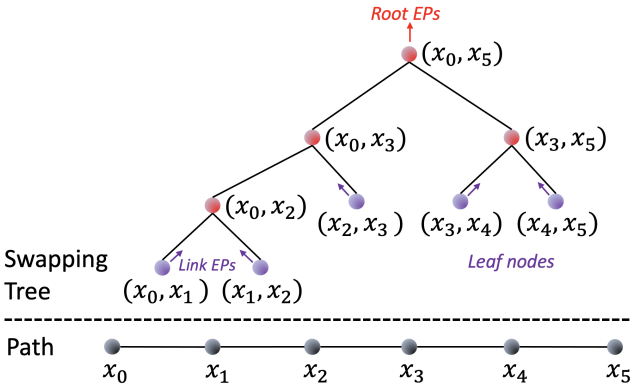


Fig. 1. A swapping tree over a path. The leaves of the tree are the link EPs, which are being generated continuously.

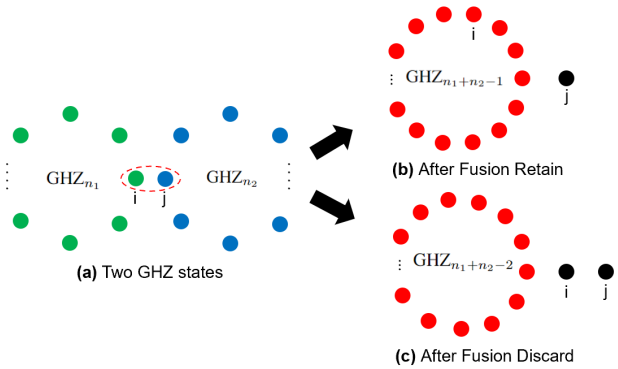


Fig. 2. Fusion Operations: Fusion-Retain and Fusion-Discard. **(a)** Two GHZ states with n_1 and n_2 number of qubits can be fused to create a bigger GHZ state, via a fusion operation over i and j qubits. To perform the fusion operation, the qubits i and j must be in the same network node. **(b)** GHZ State after Fusion-Retain; the generated GHZ state has $n_1 + n_2 - 1$ qubit with i retained and j discarded in the final state. **(c)** GHZ State after Fusion-Discard; the generated GHZ state has $n_1 + n_2 - 2$ qubits with both i and j discarded in the final state.

tion done with the same optimization objective of expected generation rate.

Our Contributions. In the above context, we make the following contributions in this paper:

1) We formulate the problem (GHZSG) of generating and distributing GHZ states with the optimization objective of maximizing generation rate under given fidelity constraints, taking into consideration the stochastic nature of quantum operations and limited network resources.
2) We design a Fusion-Retain-Only (FRO) scheme, which leverages prior work on efficient generation of remote EPs, but uses only a single type of fusion operation (§IV).
3) We also design a more general General-Fusion (GF) scheme, which uses two types of fusion operations and is based on constructing a near-optimal Steiner tree over the terminal nodes (§V).
4) Via extensive evaluations using NetSquid simulator, we show that our techniques outperform prior work and simple centralized solutions by multiple orders of magnitude while incurring a tolerable fidelity degradation (§VII).

II. Background

Generation[1] of EPs using Swapping Trees. An efficient way to generate an EP over a pair of remote network nodes (s, d) using EPs over network links is to: (i) create a path P from s to d with EPs over each of the links (i.e., adjacent nodes) on the path, and (ii) perform a series of entanglement swaps (ES) over these EPs. The series of ES operations over a given path P can be performed in any arbitrary order, but this order of ES operations affects the latency incurred in generating the EP over (s, d). One way to represent the "order" in which the ES operations are executed—is a complete binary tree over the link EPs as leaves, called a swapping tree [15]. See Fig. 1; here, the **notation** $(\mathbf{x_i}, \mathbf{x_j})$ represents an EP over a pair of qubits residing in the network nodes x_i and x_j. In some sense, x_i represents a network node as well as a qubit residing in the node, except that multiple instances of the same x_i in "different parts" of the swapping-tree represent different qubits in the same network node x_i (e.g., x_1's in the pairs (x_0, x_1) and (x_1, x_2) in Fig. 1 represents two different qubits in the node x_1. The stochastic nature of ES operations entails that generation of an EP over a remote pair of nodes using a swapping-tree may incur significant latency, called the *generation latency* (inverse of generation rate). Generation latency is largely due to the latency incurred in (i) generating the link EPs, and (ii) a generated EP (x_i, x_j) *waiting* for its "sibling" EP (x_j, x_k) to be generated before an ES operation can be performed over them to generate an EP over (x_i, x_j).

Fusion Operations to Generate GHZ States. As mentioned before, GHZ state is a type of multipartite entanglement over n qubits represented as $|\text{GHZ}_n\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n})$. GHZ states are a natural generalization of Bell states. Just like how ES operations can be used to generate an EP over remote nodes using link EPs as described above, we can use "fusion" operations to fuse two smaller GHZ states into a bigger one. These fusion operations can be used iteratively to generate bigger and bigger GHZ states, and thus, generate a desired GHZ state from link EPs. In particular, two GHZ states over m and n qubits can be fused together to generate a bigger GHZ state with $m + n - 1$ or $m + n - 2$ qubits. We use two fusion operations here: `Fusion-Retain` and `Fusion-Discard`. See Fig. 2 for a high-level description, and Figs. 3 and 4 for the detailed description of the two fusion operations. In essence, the fusion operations involve a qubit each (i and j in the figures) from the two GHZ states, and yield a GHZ state over all the qubits *except* for i (in case of `Fusion-Retain`) or both i and j (in case of `Fusion-Discard`). If qubits i and j reside in a single network node, the fusion operations will require only local (quantum) operations and classical communication and thus are LOCC methods. Our main *motivation* for using the above fusion operations is that they facilitate efficient GHZ generation algorithms. We note that the `Fusion-Retain` operation has also been used in [36] to generate GHZ states

[1]Throughout the paper, by *generation* of states, we implicitly mean generation and distribution of created states.

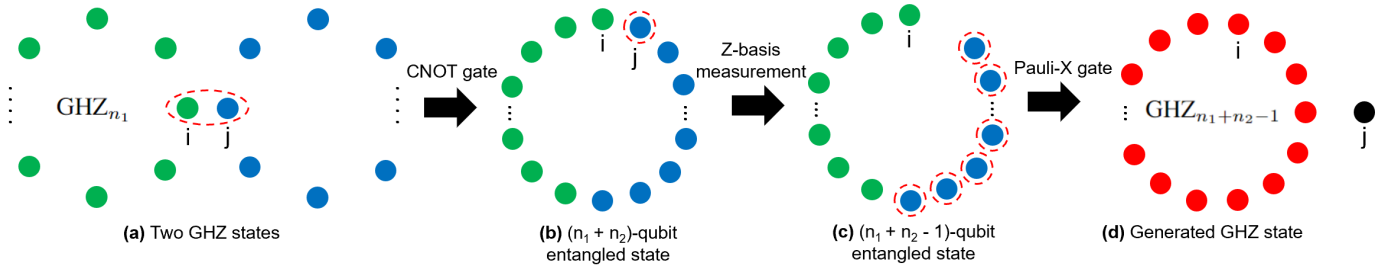


Fig. 3. Fusion Retain Operation. The depicted sequence of operations fuses an n_1-qubit with an n_2-qubits GHZ state to create a GHZ state with n_1+n_2-1 qubits. Initial GHZ States are shown in (a). **First**, a 2-qubit CNOT gate is applied on qubits i and j, to create an n_1+n_2-qubit entangled state shown in (b): $0.5(|0...0\underline{0}0\underline{0}0...0\rangle + |0...0\underline{0}1\underline{1}..1\rangle + |1...1\underline{1}0\underline{1}...1\rangle) + |1...1\underline{1}1\underline{0}...0\rangle$. We assume qubit i is the control qubit and qubit j is the target qubit, for the CNOT gate; they are underlined in the state's ket formulation. **Second**, we measure the qubit j in the Z-basis. Depending on the measurement result, the state is reduced to a different n_1+n_2-1-qubit entangled state without the j qubit (see (c)). If the j-measurement result is 0, the reduced state is $1/\sqrt{2}(|0...0\underline{0}0...0\rangle + |1...1\underline{1}1...1\rangle)$, else the reduced state is $1/\sqrt{2}(|0...0\underline{0}1...1\rangle + |1...1\underline{1}0...0\rangle)$. **Finally,** if the j-measurement result is 1, the qubits of the GHZ$_{n_2}$ apply a 1-qubit Pauli-X (NOT) gate. See (c). The final created GHZ state is a GHZ$_{n_1+n_2-1}$ state shown in (d).

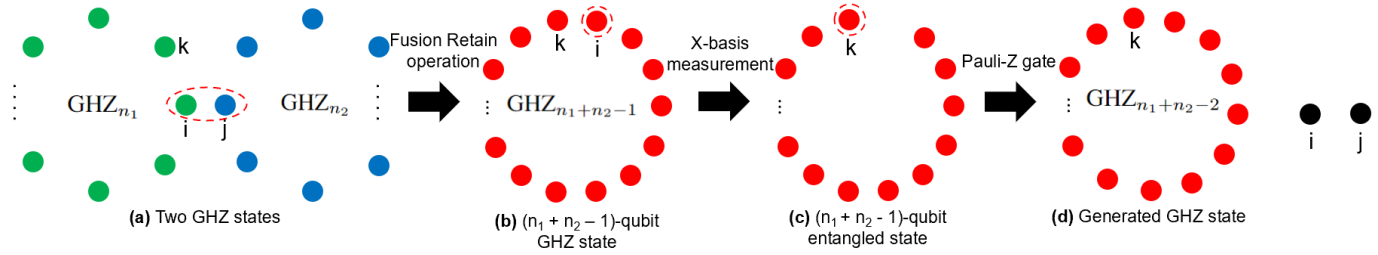


Fig. 4. Fusion Discard Operation. The depicted sequence of operations fuses an n_1-qubit with an n_2-qubit GHZ states to create a GHZ state with n_1+n_2-2 qubits. Initial GHZ states are shown in (a). **First,** a Fusion-Retain operation is performed over qubits i and j to create a GHZ$_{n_1+n_2-1}$ state with qubit i retained and j discarded. The created state (shown in (b)) can be decomposed using i qubit and be stated as: GHZ$_{n_1+n_2-1} = \frac{1}{2}(|0\rangle^{\otimes(n_1+n_2-2)} + |1\rangle^{\otimes(n_1+n_2-2)}) \otimes |+\rangle + \frac{1}{2}(|0\rangle^{\otimes(n_1+n_2-2)} - |1\rangle^{\otimes(n_1+n_2-2)}) \otimes |-\rangle$. **Second,** we measure i in the X-basis to create a n_1+n_2-2-qubit state, shown in (c). This state is either $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes(n_1+n_2-2)} + |1\rangle^{\otimes(n_1+n_2-2)})$ or $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes(n_1+n_2-2)} - |1\rangle^{\otimes(n_1+n_2-2)})$, depending on the result of measurement over i. The first is the desired GHZ$_{n_1+n_2-2}$ state, while the second state requires a correction (1-qubit Pauli-Z or phase-flip) over a random qubit k to create a GHZ$_{n_1+n_2-2}$ state. See (d).

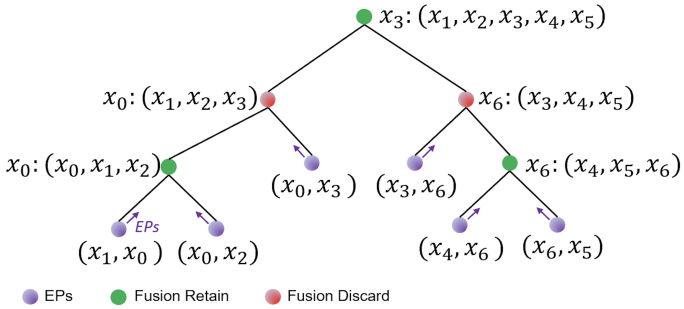


Fig. 5. Fusion Tree Example. Fusion tree over the available EPs (at the leaves of the tree) to create 5-qubit GHZ$_5$ state $(x_1, x_2, x_3, x_4, x_5)$. Here x_0 : (x_1, x_2, x_3) at a node means that the fusion operation occurred at x_0 to create a GHZ state over (x_1, x_2, x_3).

from EPs. In a related work, Clément et al. [25] propose a *Star Expansion* operation to generate GHZ states—which necessarily requires non-local gate operations unlike the above fusion operations.

Fusion Trees. Given a set of EPs, we can fuse them together using fusion operations to generate a GHZ state. In general, given a set of link EPs, one can create a GHZ state over any subset of EPs qubits using a appropriate sequence and choice of fusion operations. Similar to the concept of swapping-trees, we use a *fusion tree* to represent the order of fusion operations over given EPs as leaves. See Fig. 5. As in swapping trees, the notation $(x_1, x_2, x_3, \ldots, x_m)$ in a fusion tree represents a GHZ state over m qubits residing in m network nodes $\{x_i\}$ (one qubit per network node); in particular, the leaves of a fusion tree are pairs representing EPs over network links. Also, each interior vertex v in a fusion-tree is represented as $(x_i : X)$, where x_i is referred to as the *fusion node* while X is called the *distribution set* of network nodes; the **notation** $(\mathbf{x_i} : \mathbf{X})$ denotes a GHZ state distributed over X set of nodes created by fusing GHZ states represented by v's children (with the fusion operation taking place in the common node x_i over the two qubits in x_i one each from the two GHZ states at v's children). Each internal vertex v in the fusion tree is also colored red or green, wherein the red (green) color signifies that `Fusion-Discard` (`Fusion-Retain`) operation was used in creating the GHZ sate at v. Thus, the distribution set at a green node is a union of the distribution sets at its children, while the distribution set at a red node excludes the fusion/common node x_i.

As in the swapping trees, the stochastic nature of the underlying physical mechanisms (including, the fusion operations) means that an intermediate GHZ state may be successfully generated only after many failed attempts, and even after being successfully generated, it may need to wait for its sibling/counterpart to become available before a fusion operation can be carried out. Thus, generation of an GHZ over a remote pair of nodes may incur significant generation latency. We discuss estimation of GHZ generation latency of a fusion tree in the next section, after presenting the network model.

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 quantum network (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. Our network model is very similar to the one adopted in our recent work [15] on efficient generation of EPs. In particular, each node has an atom-photon EP generator with generation latency (t_g) and probability of success (p_g); the atom-photo generation latency implicitly includes other latencies incurred in link EP generation viz. photon transmission, optical-BSM, and classical acknowledgement. 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 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/optical BSM device in the middle of e; the optical-BSM has a certain probability of success (p_{ob}). Thus, each link is composed of two half-links with a probability of transmission success (p_e) that decreases exponentially with the link distance. To facilitate atom-atom ES and fusion operations, each network node is also equipped with an atomic-BSM device with BSM (fusion) operation latency of t_b (t_f) with a probability of success p_b (p_f). There is an independent classical network with a transmission latency of t_c; we assume classical transmission always succeeds.

GHZ State Generation Latency Using a Fusion Tree. Fundamentally, the structure of fusion trees is similar to that of swapping trees, and hence, the technique to estimating generation latency (inverse of generation rate) of a swapping tree from [15] can be directly applied here. Thus, for a non-leaf vertex t in a fusion tree with children/subtrees t_l and t_r, the generation latency of the GHZ state corresponding to the vertex t can be estimated as:

$$L_t = (\frac{3}{2}\max(L_l, L_r) + t_f + t_c)/p_f, \tag{1}$$

where L_l and L_r are the generation latencies of the GHZ states corresponding to the children t_l and t_r, and t_f, t_c, and p_f are the network parameters as defined above. The above assumes that the generation latencies of the subtrees t_l and t_r are exponentially distributed. Generation latency of a leaf in a fusion tree is given by the generation latency of the EP at the leaf–which may in turn be the generation latency of the swapping tree used to generate the EP. To estimate the overall generation latency of a fusion tree, we apply the above recursive equation iteratively, while assuming that the resulting latencies also have an exponential distribution.

A. Problem Formulation

In this section, we formulate the problem of efficiently generating GHZ state across an arbitrary set of nodes.

GHZ State Generation (`GHZSG`) Problem. Given a quantum network and a set of *terminals* $T = (t_1, t_2, ..., t_m)$, the `GHZSG` problem is to determine a fusion tree $\mathcal{F}$ that generates a GHZ state over the given terminals with minimum expected generation latency under the following constraints.

1) *Node Constraints.* For each node, the aggregate resources used by $\mathcal{F}$ is less than the available resources; we formulate this formally below.
2) *Fidelity Constraints.* The fusion tree should satisfy the following: (a) Number of leaves[2] is less than a given threshold τ_l; this is to limit fidelity degradation due to gate operations. (b) Total memory storage time of any qubit is less than a given *decoherence threshold* τ_d.

Formulating Node Constraints. Consider a fusion tree $\mathcal{F}$, and let $\mathcal{E}$ be the set of all links involved in the generation of EP at the leaves of $\mathcal{F}$. Note that a leaf of $\mathcal{F}$ may represent EPs generated by a swapping tree over a path P, in which case all the links in P should be included in $\mathcal{E}$. Now, for each link $e \in \mathcal{E}$, let $R(e, \mathcal{E})$ be the "total" EP rate being used by $\mathcal{F}$ over the link e. Let us define $E(i)$ be the set of links from $\mathcal{E}$ incident on node i. Then, the node capacity constraint is formulated as follows.

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

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, and that $1/t_g$ is a node's total generation capacity. Also, the memory constraint is that for any node i, the memory available in i should be more than $|E(i)|$.

B. Related Works

There has been recent interest in schemes for efficient generation of GHZ states in a quantum network. These works have largely focused on the objective of minimizing the *number* of link EPs consumed in generating and distributing the desired GHZ states. Such approaches can be categorized into centralized and distributed generation schemes, as discussed below. To the best of our knowledge, there have been no prior work on efficient generation and distribution of GHZ states in terms of optimizing generation latency/rate while taking into considering the stochastic nature of underlying processes/operations—this is the focus of our work.

Centralized Generation Schemes. In a centralized generation scheme, an appropriately chosen central node first creates the desired GHZ locally, and then teleports qubits to the desired terminal nodes using EPs between the central and the terminal nodes. Bugalho et al. [29] propose an optimal way of selecting the best central node for 3-qubit GHZ states. For larger GHZ states, the proposed approach is generalized but without any optimality guarantees. Aves et al. [30] focus on latency and fidelity analytical formulation for different generation schemes in a star network graph where terminals are connected to a central node via homogeneous quantum channels. Finally, [28] proposes a max-flow based approach to minimize the number

[2]Here, the number of leaves includes the leaves of any swapping trees used to create the EPs at the leaves of the fusion tree.

of EPs consumed in generating and distributing a GHZ state using a centralized generation scheme. They represent the teleportation routes as multi-path flows in the network, and use a network flow approach to maximize the total teleportation rate and thus the GHZ generation rate. The network-flow approach allows representation of network resource constraints—e.g., how a single network link can be used across multiple teleportation routes. However, the flow representation ignores the stochastic aspect of the teleportation (or entanglement-swapping) process which fundamentally requires taking into consideration the length of the teleportation paths (ignored in the network-flow representation).

Distributed Generation Schemes. In a distributed generation scheme, the target GHZ state is generated in a distributed manner (perhaps, by iteratively merging smaller GHZ states)—as in the schemes discussed in this paper. In [25], the authors propose a *star expansion* operation/sub-protocol to fuse together EPs, and use the operation iteratively to generate the desired GHZ state. They design an algorithm that minimizes the number of EPs consumed to create the desired GHZ state; we compare our schemes to their approach in §VII. In [31], the authors introduce a 2D repeater chain wherein GHZ states over smaller distances are fused to form GHZ states at larger distances, analogous to how EPs over remote pairs of nodes are created using entanglement-swapping operations over a path. However, the proposed approach is only applicable to special network topologies. Finally, [32] proposes an approach to determine the best sequence of fusion operations and measurements over given EPs to create a GHZ state of highest fidelity.

Our Approach: Minimizing Generation Latency. The goal of our work is to develop schemes that optimize the expected generation latency (and hence, generation rate) of desired GHZ states distributed over a given set of network nodes, under fidelity constraints. Accurate estimation of generation latency requires incorporating the stochastic nature of the underlying processes and operations, which has been largely ignored in the prior works described above. However, for the simpler and special case of generation of EPs over remote pairs, there has been considerable work done on developing schemes that minimize generation latency. E.g., Shi and Qian [33] and Chakraborty et al. [34] design algorithms to select the best entanglement routing paths, and [15] develops an optimal swapping tree for a pair of remote nodes in the network.

IV. Fusion-Retain-Only (FRO) Algorithm

In this section, we design a GHZ generation algorithm that uses only the `Fusion-Retain` fusion operations.

Basic Idea. To create a GHZ state over given terminal nodes, one approach can be to first create EPs (smallest GHZ states) over *some* of the terminal pairs, and then iteratively create bigger and bigger GHZ states till the final desired GHZ state is created. Here, beyond the generation of EPs, the only nodes involved in the fusion operations are the given terminal nodes, and thus, we only need to use the `Fusion-Retain` fusion operations. Thus, we refer to this approach as the `Fusion-Retain-Only` (`FRO`) algorithm. The main motivation for the above approach is to be able to leverage known efficient algorithms [15] for generation of EPs. At a high-level, the `FRO` algorithm consists of the following three steps (see Fig. 6).

1) Selection of a set of terminal-**pairs** $\mathcal{T}$ for EPs generation.
2) Efficient and *simultaneous* generation of EPs over each of the terminal-pairs in the selected set $\mathcal{T}$.
3) Computing an efficient fusion tree to generate the desired GHZ state from the EPs over the terminal-pairs in $\mathcal{T}$.

We develop optimal or near-optimal algorithms for each of the above steps, as described below.

1. Selection of Terminal-Pairs $\mathcal{T}$. In this first step, we want to select a set $\mathcal{T}$ of terminal-pairs over which to generate EPs that can be fused to compute the desired GHZ state. The set of terminal-pairs $\mathcal{T}$ should be such that they are "connected", i.e., they form a connected tree in a complete graph over the set of given terminals T, since we need to fuse the generated EPs to generate a GHZ state over T. In addition, we want the aggregate generation latency of the EPs over the chosen $\mathcal{T}$ to be small—so as to minimize the overall generation latency of the final GHZ state.

Based on the above conditions, the problem of selection of a set of terminal-pairs $\mathcal{T}$ can be defined as that of selecting a minimum-weighted spanning tree (MST) in a weighted complete graph over terminals. More formally, consider the *complete* graph $G_T = (V = T, E = (T \times T))$ over the set of terminals T. We associate each edge $e = (t_i, t_j)$ with a weight w_e equal to the EP generation latency (see below) over the terminal-pair (t_i, t_j). Our problem's objective is to select a set $\mathcal{T}$ of terminal-pairs that form a connected tree in G_T and have the minimum sum of weights, i.e., minimum *sum* of latencies of the EPs over the chosen $\mathcal{T}$.

This objective is equivalent to determining a minimum spanning tree in G_T, which can be optimally computed using well-known techniques [37]. We determine the weight w_e, i.e., the (independent) EP generation latency over a pair of terminals $e = (t_i, t_j)$, based on the optimal EP generation scheme in [15] which computes the optimal swapping tree (for generation of EP) over each pair of network nodes. Note that in practice the EPs will be generated *simultaneously*, and hence, the objective of *sum* of generation latencies of pairs in $\mathcal{T}$ is not the perfect objective—but we have chosen the above sum-of-latencies objective for the sake of simplicity and to facilitate design of an efficient algorithm. We discuss efficient generation of simultaneous EPs in the next paragraph.

2. Efficient Generation of EPs over Pairs in $\mathcal{T}$. As mentioned above, the weights associated with the edges in G_T are the *independent* generation latencies, i.e., when the terminal-pair is generating EPs by itself in the quantum network. However, in our context, we *can* generate the EPs over $\mathcal{T}$ *simultaneously*; this simultaneous generation of EPs over multiple pairs entails sharing of network resources (e.g., node generation capacities). For efficient simultaneous generation

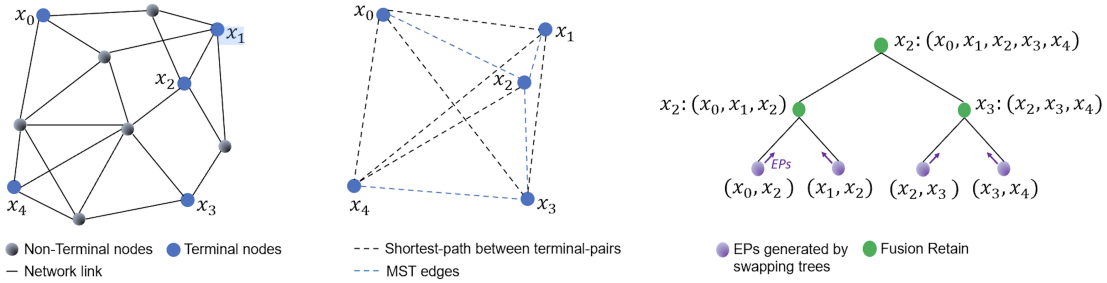


Fig. 6. Fusion-Retain Only (FRO) Algorithm. (a) Network graph with terminals over which a GHZ state needs to be created. (b) Complete graph over the terminals, with each terminal-pair x's weight equal to the EP generation latency of EPs over the terminal-pair x. The red-dashed edges forms the minimum spanning tree (MST) in this weighted graph. (c) Fusion tree computed over the MST edges, to create a GHZ state over the terminal nodes. Note that the leaves of the fusion tree are pairs of, possibly remote, nodes storing the EPs; if a node-pair is remote then the EPs are generated using swapping trees.

of EPs, we use an optimal linear-program (LP) developed in [15] which incorporates node capacity constraints. Given a quantum network and a set of node-pairs ($\mathcal{T}$ in our case) over which to generate EPs, the LP developed in [15] yields a "flow" of entanglement paths that generate EPs over the given set of pairs $\mathcal{T}$ such that the aggregate generation rate is maximized. We note that the LP approach, even though optimal for a given set of terminal-pairs, can't be used easily for *selection* of terminal-pairs that maximize the LP objective.

3. Optimal Fusion Tree over Generated EPs. We now need to determine an efficient fusion tree to generate the desired GHZ state over the EPs being generated over pairs in $\mathcal{T}$. Recall that a fusion tree essentially determines the order in which the initial EPs and the intermediate GHZ states are fused, and thus, determines the overall GHZ generation latency (see §III). A simple approach to construct a reasonably efficient fusion tree could be to create a "balanced" binary tree over the given pairs in $\mathcal{T}$, with each interior node representing a `Fusion-Retain` operation over its two children. However, since the EPs generation latencies over the terminal-pairs in $\mathcal{T}$ may not be equal, a balanced fusion tree may not yield an optimal GHZ generation latency. In our context, we can actually construct an *optimal* fusion tree as follows.

Optimal Dynamic Programming (DP) Algorithm. The recursive expression for GHZ generation latency (§II) suggests a DP approach to compute an optimal fusion tree. Recall that G_T is a complete graph over terminals, and $\mathcal{T}$ is a selected set of terminal-pairs (edges in G_T) which form a connected spanning tree in G_T. For a general subset of terminal-pairs $\mathcal{T}'$ ($\subseteq \mathcal{T}$) that form a connected tree (but not necessarily spanning) in G_T, let $L[\mathcal{T}']$ be the optimal latency for generating a GHZ state over the terminals $\mathcal{T}'$ using a fusion tree that uses only `Fusion-Retain` operations over $\mathcal{T}'$. Based on Eqn. (1), we can easily derive $L[\mathcal{T}']$ recursively as follows.

$$L[\mathcal{T}'] = (\frac{3}{2}B + t_c + t_f)/p_f \tag{3}$$
$$\text{where} \quad B = \min_{\mathcal{T}'_l \subseteq \mathcal{T}'} \quad \max\left(L[\mathcal{T}'_l], L[\mathcal{T}'_r = \mathcal{T}' - \mathcal{T}'_l]\right)$$

Above, $\mathcal{T}'_l$ must be connected in G_T, which also implies that $\mathcal{T}'_l$ and $\mathcal{T}'_r$ will have exactly one node in common. The above equation can be used to compute $L[\mathcal{T}']$ (and associated optimal fusion tree) for all $\mathcal{T}' \subseteq \mathcal{T}$, and thus, to compute $L[\mathcal{T}]$ and the associated optimal fusion tree over $\mathcal{T}$, using a DP algorithm. The above DP algorithm is actually exponential in number of terminals, which is acceptable since the number of terminals is expected to be small (e.g., at most 10). Note that the size of $\mathcal{T}$ is equal to the number of terminals (and not square of the number of terminals) as the pairs in $\mathcal{T}$ form a tree.

Overall Performance Guarantee and Fidelity Constraints. We note that the overall FRO algorithm doesn't have a provable performance guarantee, but each of the three steps as defined are provably optimal. For clarity of presentation, we have ignored the fidelity constraints in the above description, but incorporate them later in §VI. Note that the node constraints are incorporated in the LP of the second step.

V. General Fusion (`GF`) Algorithm

We now design the `General Fusion (GF)` algorithm, which uses both fusion operations, and thus, is more general than the `FRO` algorithm. For clarity, we ignore the node and fidelity constraints for now, and incorporate them later in §VI.

Basic Idea. The `General Fusion (GF)` algorithm, like the previous `FRO` algorithm, also builds bigger and bigger GHZ states from smaller GHZ states until the desired GHZ state is created—but, unlike the `FRO` algorithm, it may also involve the non-terminal nodes beyond the EPs generation and hence may use `Fusion-Discard` operations too. Thus, `GF` is fundamentally a more general (i.e., less restrictive) algorithm than the `FRO` algorithm, and hence, has the potential to outperform the `FRO` algorithm if designed effectively. At a high level, the `GF` algorithm first selects a set $\mathcal{E}$ of network links (node pairs) over which to generate the link EPs, and then, constructs an efficient fusion tree over $\mathcal{E}$. Since the EPs over $\mathcal{E}$ must yield (via fusion operations) a GHZ state over the given terminals, the set of links $\mathcal{E}$ must form a connected graph and include the terminals. Thus, the `GF` algorithm consists of the following two high-level steps (see Fig 7).

1) Select the "best" Steiner tree $\mathcal{S}$ in the original network graph over the given set of terminals T. Here, the Steiner tree is essentially a subgraph (not necessarily induced) that is connected and includes the terminals.
2) Construct an efficient fusion tree over the links/edges in $\mathcal{S}$. The fusion tree essentially generates GHZ states from the EPs over the links in $\mathcal{S}$.

Below, we develop a near-optimal algorithm for the first step and an efficient heuristic for the second, as described below.

1. Selecting the "Best" Steiner Tree. Here, we develop an optimal algorithm to find the best Steiner tree $\mathcal{S}$ in the given

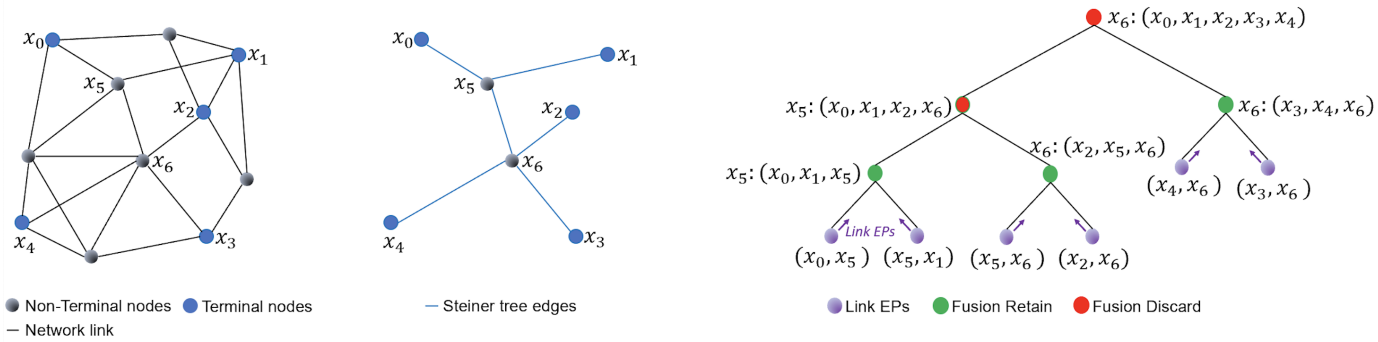


Fig. 7. General Fusion (GF) Algorithm. (a) Network graph with terminals over which a GHZ state is to be created. (b) Best Steiner tree computed over the terminals. (c) Fusion tree over the Steiner tree edges, to create the GHZ state over the terminals. (Fig. 8 shows how to go from (b) to (c), i.e., how to create a fusion tree from a Steiner tree).

network graph, based on an appropriately define objective function. In graph theory, a Steiner tree in a given graph $G = (V, E)$ for a set of terminals $V' \subseteq V$ is a subgraph (not necessarily induced) in G that is a tree and contains all the terminals in V'. In our context, we want to construct a Steiner tree $\mathcal{S}$ in the network graph that includes the set of terminals T. Ideally, the optimization objective should be to construct a Steiner tree that yields a fusion tree of minimum GHZ generation latency over its links. Such an optimization objective is however intractable to optimize, since it is even intractable to compute an optimal fusion tree over a *given* Steiner tree. Thus, we first derive a simpler optimization objective for a Steiner tree, which is simple as well as reflective of the GHZ generation latency of an efficient fusion tree over the Steiner tree.

Optimization Objective. To define an optimization objective for a given Steiner tree $\mathcal{S}$, we need to first estimate the GHZ generation latency of an efficient fusion tree, that we can design an algorithm to construct, over the links in $\mathcal{S}$. As motivated and described in the second step later, we use a a *balanced* binary[3] tree over the links in $\mathcal{S}$ as the fusion tree to generate the desired GHZ state. Thus, we define the optimization objective for a Steiner tree $\mathcal{S}$ by estimating the GHZ generation latency of an appropriate balanced fusion tree over $\mathcal{S}$. Consider a given Steiner tree $\mathcal{S}$ and a "balanced" fusion tree $\mathcal{F}$ over it. We define the objective function $B(\mathcal{S})$ as the GHZ generation latency due to $\mathcal{F}$ which can be derived as follows.

$$B(\mathcal{S}) = p^d N + [(p^d - 1)/(p - 1)](t_f + t_c)/p_f \qquad (4)$$

where $p = 3/(2p_f)$ and d and N are defined as follows. Let x be the link with highest generation-latency in $\mathcal{S}$; then, N is x's generation latency and d is its depth in the balanced fusion tree $\mathcal{F}$. For a balanced fusion tree, it is easy to see that d is either $\lceil(\log_2 |\mathcal{S}|)\rceil$ or $\lfloor(\log_2 |\mathcal{S}|)\rfloor$, but for simplicity we assume d to be just $\lceil(\log_2 |\mathcal{S}|)\rceil$ to avoid defining $B(\mathcal{S})$ in terms of the exact structure of $\mathcal{S}$.

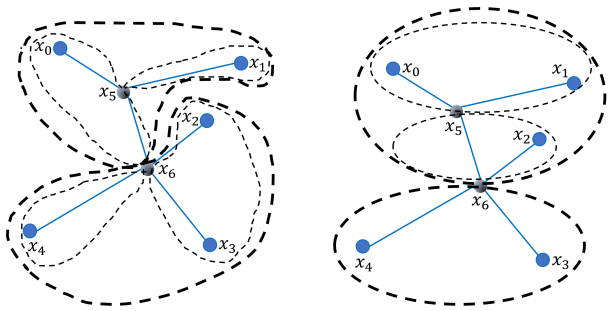
Selecting a Near-Optimal Steiner Tree. Note that the above objective function for a Steiner tree $\mathcal{S}$ depends on only two parameters of $\mathcal{S}$: (i) Number of links in $\mathcal{S}$, and (ii) The maximum EP generation latency of a link in $\mathcal{S}$. In addition, it is easy to see that, for a *given/fixed* maximum edge-latency, a Steiner tree's objective value decreases with the decrease in the number of links in it. Thus, for each potential value of maximum edge-latency l, we construct a subgraph G_l, of original network graph, consisting of links with EP generation latencies lower than l, and then construct a minimum-size Steiner tree S_l in G_l. Since there are only a polynomial number of links and thus only a polynomial number of potential l values, we can do the above for each potential l value and pick the best Steiner tree S_l. To compute the minimum-size Steiner tree in a given G_l, we can use a well-known 2-approximate algorithm [38]. The above overall algorithm thus constructs a Steiner tree whose objective value is at most 2 times the optimal objective value.

2. Balanced Fusion Tree Over the Steiner-Tree Links. Having constructed a Steiner tree $\mathcal{S}$ over the given terminals, we now need to create an efficient fusion tree over the links in $\mathcal{S}$—to create the desired GHZ states from the generated EPs over the links, using fusion operations. The exponential-time DP algorithm used to construct a fusion tree in the `FRO` algorithm is not feasible here, since the number of links in our Steiner tree can be much larger than the number of terminals–and in general, can be as large as the quantum network size. Thus, here we design an efficient polynomial-time heuristic based on a divide-and-conquer strategy. Our proposed heuristic consists of the following three high-level steps.

1) Divide the given Steiner tree $\mathcal{S}$ with terminals T into two near-disjoint (with only one node c in common) and approximately "equal" subtrees $\mathcal{S}_r$ and $\mathcal{S}_l$. We consider two measures of equality—equal in size or in estimated generation latency—and discuss these measures and schemes in detail below.
2) Recursively compute a fusion tree $\mathcal{F}_r$ ($\mathcal{F}_l$) for the subtree $\mathcal{S}_r$ ($\mathcal{S}_l$). The fusion tree $\mathcal{F}_r$ ($\mathcal{F}_l$) should be such that the generated GHZ state is over the set of terminals from $\mathcal{S}$ in $\mathcal{S}_r$ ($\mathcal{S}_l$) and the common node c.[4]
3) Create a fusion tree with a root R and its two children as $\mathcal{F}_r$ and $\mathcal{F}_l$. The root R represents a fusion operation at the node c common to the subtrees $\mathcal{S}_r$ and $\mathcal{S}_l$. If $c \in T$, then a `Fusion-Retain` operation is used at c, else a `Fusion-Discard` operation is used.

[3]Note that our fusion operations are binary, and hence, the fusion trees in our context are binary trees.

[4]Note that subtree is associated with a set of subtree-terminals of its own, which may include nodes not in the original set of terminal T.



(a) Equi-Size division. **(b)** Equal-Latency division.

Fig. 8. Two schemes to recursively divide a Steiner tree to create a fusion tree. **(a)** Equi-Size Division. The Steiner tree is recursively divided into two almost equal-size subtrees. **(b)** Equal-Latency Division. The Steiner tree is recursively divided into two subtrees with similar estimated generation latencies; this partitioning yields the fusion tree shown in Fig. 7(c).

Division into Equi-Size Subtrees. Dividing a tree into two equal-sized subtrees may not be straightforward—e.g., consider a star tree. Note that in our context we need to have one node in common between the subtrees, to facilitate a fusion operation over the GHZ states generated from the subtrees. Our heuristic considers each possible common node c, and considers dividing the tree into almost-equi-sized subtrees with c as the common node. Consider a node c in the given tree, and let c's degree be m. Let $s_1, s_2, \ldots, s_m$ be the sizes of the subtrees of c when we visualize the given tree as rooted at c. To create almost-equal division of the given tree into two subtrees with c as the common node, we need to just partition the set $\{s_i\}$ into two subsets of almost equal sum; this is the well-known NP-hard `Partition` problem which has a well-known pseudo-polynomial optimal algorithm based on dynamic programming. We repeat the above for each node c in the tree, and pick the best division into sub-trees; here, the best division is the one with minimum difference in the sub-tree sizes. See Fig 8(a).

Equal-Latency Division. To divide a tree into two subtrees that would yield fusion trees of almost-equal generation latencies, we use a similar approach as above—except for the calculation of the s_i values. For equi-latency division, s_i should be an estimate of the generation latency of the envisioned/optimal fusion tree over the i^{th} subtree. Thus, to determine s_i, we assume the fusion tree to be a balanced binary tree, and then, use Eqn. 4 to estimate the GHZ generation latency. See Fig 8(b).

VI. Incorporating Node and Fidelity Constraints

We now discuss changes to the algorithms to incorporate the node and fidelity constraints of the `GHZSG` problem formulation in §III-A.

Node Constraints. Recall that the node constraints are incorporated in the LP of the second step of the `FRO` algorithm. In addition, the node-constraints are also somewhat incorporated in the first step of the `FRO` algorithm, by assigning weights to terminal-pairs based on generation latency using an optimal swapping tree which incorporates node-constraints for each pair of nodes [15] . We incorporate node constraints in the `GF` algorithm as follows. In the first step of the `GF` algorithm, where we compute the best Steiner tree, we incorporate the node constraints as in the `FRO`'s first step—i.e., by assignment weights to links based on generation latency using an optimal swapping tree [15]. In addition, in the second step of the `GF` algorithm, wherein we compute a balanced fusion tree based on equi-*latency* division, we can further incorporate node constraints in determining the maximum generation latency of a link in a subtree.

Fidelity Constraints. We now discuss how to incorporate the fidelity constraints as defined in the `GHZSG` problem formulation in §III-A. Recall that the fidelity constraint is represented as a combination of (i) memory storage time (age) threshold of τ_d, and (ii) number of leaves (operations-related fidelity) threshold of τ_l. We discuss these thresholds below.

Enforcing Age Threshold τ_d. We start with discussing how we can incorporate the constraint of the expected age of any qubit to be at most τ_d during generation of a GHZ state, while constructing the fusion tree.

In the **FRO scheme**, the age of a qubit comes from two processes: (i) EP generation for a terminal-pair using a swapping tree, and (ii) GHZ generation from EPs using a fusion tree. We divide the total age threshold τ_d into two thresholds, viz., τ_d^s and τ_d^f, for swapping tree and fusion tree respectively, based on the relative heights of an average swapping tree and the fusion tree. The swapping-tree age threshold τ_d^s can be enforced in determining the swapping trees as per [15], while the fusion-tree age threshold τ_d^f can be enforced while constructing a fusion tree as follows. Recall that the `FRO` algorithm uses a DP algorithm to construct a fusion tree. Enforcing age threshold in the DP algorithm is relatively straightforward, by including an additional threshold-parameter in the $L[\,]$ function (see Eqn. 3) and filtering out fusion trees that violate the threshold parameter; determination of the threshold-parameter of subtrees from that of a parent's node can be done as in [15].

In the **GF scheme,** a qubit ages purely due to GHZ generation using a fusion tree from link EPs. Since we consider only balanced fusion trees in the second step of `GF`, we can transform the age threshold approximately into a threshold on the height of the fusion tree—which is, in turn, a function of the size of the Steiner tree computed in the first step. The size of the Steiner tree can be easily constrained in the Step 1's algorithm, at the cost of a possibly higher generation latency of the bottleneck link.

Enforcing Operation Threshold τ_l. For the **FRO scheme**, the τ_l threshold can be enforced similar to the above age threshold, by dividing the threshold into swapping and fusion trees, handling swapping-tree threshold over number of leaves as per [15], and finally, enforcing the number of leaves constraint in the fusion tree by restricting its height in the DP approach. For the **GF scheme**, enforcing the τ_l threshold essentially amounts to restricting the number of leaves of the fusion tree, which can be easily done by restricting the size of the Steiner tree as we only consider balanced fusion trees.

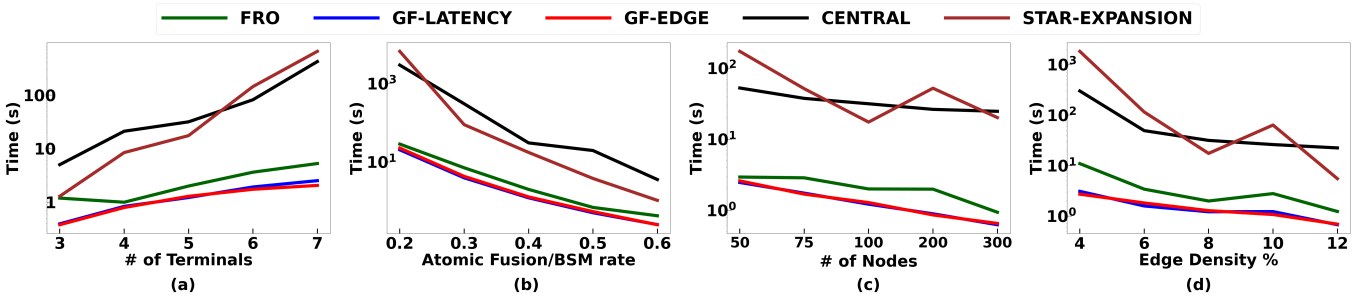


Fig. 9. GHZ state generation latency for varying parameters. (a) Varying number of terminals. (b) Varying fusion success rate. (c) Varying number of network nodes. (d) Varying network edge density. Here, for some data points (e.g., 0.2 and 0.3 fusion/BSM rates in (b)), the generation latency of `Central` and `SE` was more than our simulation duration of 100 seconds, and thus, we have conservatively used latency values from analytical results for these schemes, for the sake of completeness.

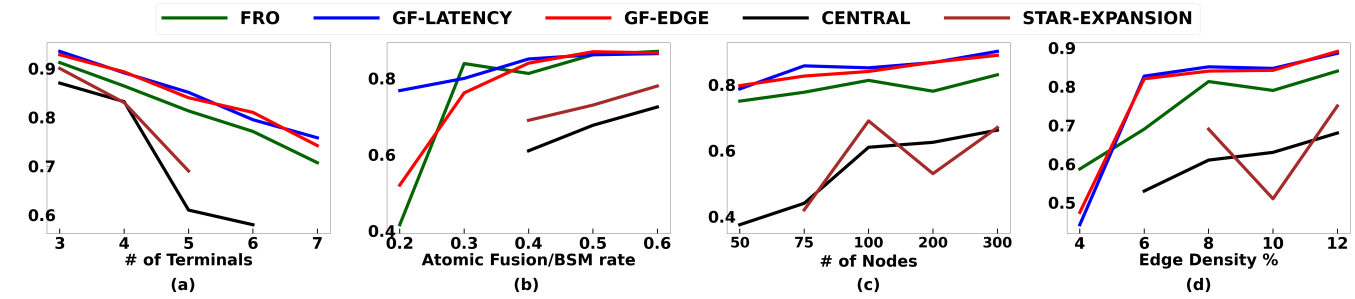


Fig. 10. GHZ State fidelity for varying parameters. (a) Varying number of terminals. (b) Varying fusion success rate. (c) Varying number of network nodes. (d) Varying network edge density. Here, for some data points (e.g., 0.2 and 0.3 fusion/BSM rates in (b)), the generation latency of `Central` and `SE` schemes was more than our simulation duration of 100 seconds, and thus, these data points are missing for these schemes.

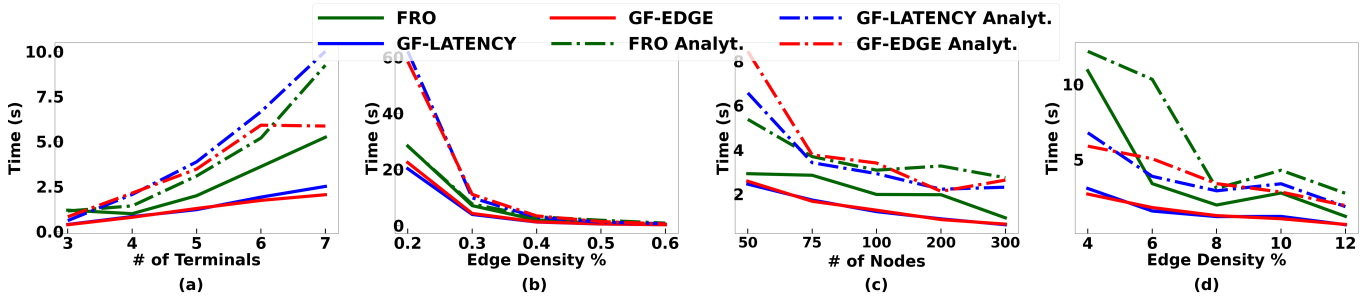


Fig. 11. Comparing analytical results vs. simulation results in terms of GHZ state generation latency for varying parameters. (a) Varying number of terminals. (b) Varying fusion success rate. (c) Varying number of network nodes. (d) Varying network edge density.

VII. Evaluations

Here, we evaluate the generation latency and fidelity of generated GHZ states by our developed schemes and compare them with a prior work and a simple/naive approach. We also validate the accuracy of our analytical models by comparing the analytical results with those generated using a quantum network simulator. We implement our schemes over the discrete event simulator for QNs called NetSquid [39].

GHZ Generation Protocol. Our algorithms compute a fusion tree over link-level or network-level EPs, and we need a way to implement it on a quantum network. We build our protocols on top of the link-layer protocol of [40], which is delegated with the task of continuously generating EPs on a link at a desired rate (as per the fusion tree specifications). For the `FRO` algorithm, we use the swapping tree protocol of our prior work [15] to generate EPs over the selected terminal-pairs. As the links (in `GF`) or terminal-pairs (in `FRO`) generate EPs, we need a protocol to fuse either the EPs or smaller GHZ states using the selected fusion tree. Omitting the tedious bookkeeping details, the key aspect of our GHZ-generation protocol is that fusing operation corresponding to node x in the fusion tree is done only when both the sub-GHZ states (corresponding to the subtrees of x in the fusion tree) have been generated. We implement all the gate operations (including, atomic and optical BSMs, fusion operations) within NetSquid to keep track of the fidelity of the generated GHZ states. On **success** of a fusion operation, the fusion node transmits classical information to the terminal nodes of both sub-states, to manipulate their qubits or to start the next-level fusion operations. More specifically, the fusion node sends one classical bit of result to all the terminals in the right subtree, and a classical Success-ACK bit to the terminals located in the left subtree. The success-ACKs serve as notifications to the nodes to start next-level fusion operations, if any. On **failure** of a fusion operation, a classical Failure-ACK is sent to all descendant leaf-nodes, so that they can now start generating new link EPs.

Simulation Setting. We generate random quantum networks in a similar way as in the recent works [15], [33]. By default, we use a network spread over an area of $100km \times 100km$. We

use the Waxman model [41], used to create Internet topologies, to randomly distribute the nodes and create links; we use the maximum link distance to be 10km. We vary the number of nodes from 50 to 300, with 100 as the default value. We choose the two parameters in the Waxman model to maintain the number of links to 8% of the complete graph (to ensure an average degree of 3 to 15 nodes). We select the desired number of terminal nodes randomly within the network graph. Each data point in the plots is an average of 10 random simulations, each of a duration of 100 seconds.

Parameter Values. We use parameter values mostly similar to the ones used in [15], [35], and vary some of them. In particular, we use fusion probability of success (p_f) to be 0.4 and latency (t_f) to be 10 μ secs; in some plots, we vary p_f from 0.2 to 0.6. The values of atomic-BSM probability of success (p_b) and latecny (t_b) are always equal to their fusion counterparts p_f and t_f, respectively. This is done so `FRO` and `GF` schemes be comparable. The optical-BSM probability of success (p_{ob}) is half of p_b. We, for generating link-level EPs, use atom-photon generation times (t_g) and probability of success (p_g) as 50 μsec and 0.33 respectively. We use the size of the GHZ state, number of terminals, to be 5; we vary it from 3 to 7, in some plots. Finally, we use photon transmission success probability as $e^{-d/(2L)}$ [35] where L is the channel attenuation length (chosen as 20km for an optical fiber) and d is the distance between the nodes.

Fidelity is modeled in NetSquid using two parameter values, viz., depolarization (for decoherence) and dephasing (for operations-driven) rates. As in [15], we choose a decoherence time of two secs based on achievable values with single-atom memory platforms [42]; note that decoherence times of even several minutes [43], [44] to hours [45], [46] has been demonstrated for other 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% [34].

Algorithms and Performance Metrics. We evaluate our following algorithms: `FRO`, Edge-Balanced `GF` (`GF-Edge`), and Latency-Balanced `GF` (`GF-Latency`). For comparison, we also evaluate the `Star Expansion` (`SE`) algorithm from [25], and a simple approach called `Central`. We describe these below. The `Star Expansion` (`SE`) [25] algorithm essentially computes a Steiner tree S over the network graph, and iteratively and sequentially, performs a *star-expansion* operation over: (i) a node l that is a terminal as well as a leaf in S; this node l remains fixed through all iterations; and (ii) a randomly-chosen "neighbor" (in this iteration) of l. In [25], the author assume the fusion star-expansion operation to be deterministic in optimizing their objective of the number of EPs consumed. In our evaluation of `SE`, we conservatively use the probability of failure of the star-expansion operation involving d qubits to be $qd/2$, where q is the probability of failure for our binary-fusion operations. The `Central` approach works by first generating the final GHZ state locally and then teleporting the qubits of the GHZ state to the desired terminals. More formally, the `Central` node picks a "central" node (as described later) C, generates the desired GHZ state at C, generates EPs between C and each of the terminals, and then teleport the GHZ qubits to the terminals using the generated EPs. To continuously generate the GHZ states at an optimal generation rate, the generation of EPs between C and the terminals is done continuously in parallel with other steps. To generate the EPs over C and each of the terminals simultaneously, we use the optimal linear programming approach from [15]. We determine the best central node using an exhaustive search over all the nodes in the network, and picking the one that yields the minimum estimated GHZ generation latency.

`GHZSG` **Results.** Fig.9 plots the GHZ state generation latency for various schemes for varying number of terminals, number of network nodes, success probability of BSM/fusion operations, and network link density. We observe that `GF-Latency` and `GF-Edge` schemes perform similarly and they both outperform `FRO` by a good margin, and `Central` and `SE` schemes by an order of magnitude. Best performance of `GF` schemes validate use of non-terminal nodes for fusion operations and our approaches for balanced fusion trees. The `Central` scheme performs very bad since in this scheme all the teleportation operations from the central node to the terminals need to succeed, else the whole process must starts from scratch as the GHZ state is destroyed. Similarly, the `SE` schemes performs very bad due to the fact that the star-expansion fusion operations are performed in a *sequence*, which essentially results in a skewed fusion tree of a large height resulting in very high latency as fusion operations are probabilistic. The `Central` as well as `SE` approaches sometimes even have a generation latency of more than 100 seconds, our simulation duration; for these cases, we plot the generation latency computed analytically.

Fidelity Results. We now investigate the fidelity of the GHZ state generated. See Fig. 10. We observe that `GF-Latency` and `GF-Edge` schemes perform the best as there is expectedly a high correlation between the generation latency and the time a qubit stays in a quantum memory. `FRO` scheme also produces GHZ states with reasonable fidelity but underperforms the other two schemes as the spanning tree over terminal-pairs is conceptually built over swapping trees for the terminal-pair which increases the total number of link involved compared to the `GF` schemes which is based on Steiner tree connecting the terminals directly. As mentioned above, the `Central` and `SE` schemes sometimes have generation latencies of more than the simulation duration, which results in no GHZ states being produced—thus, Fig.10 plots have some missing data points for these scheme.

Validating the Analysis. Finally, we compare the generation latency computed from the simulations to those estimated analytically (as per Eqn. (1)). See Fig. 11. We observe that the latencies from the two approaches match closely, which validates our analytical methodology and assumptions therein.

VIII. Conclusions

We have designed techniques for efficient generating and distribution of GHZ states, taking into consideration the stochastic nature of quantum operations. Our future work is focused on exploring more sophisticated approaches by incorporating purification techniques to further improve fidelity and effective generation rate of such high-fidelity GHZ states.

Acknowledgment

This work was supported by NSF awards FET-2106447 and CNS-2128187.

References

[1] F. Arute et al, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505—510, 2019.

[2] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, "Quantum internet: From communication to distributed computing!" in *5th ACM International Conference on Nanoscale Computing and Communication*, 2018.

[3] C. Simon, "Towards a global quantum network," *Nature Photonics*, vol. 11, no. 11, pp. 678–680, 2017.

[4] Z. Eldredge, M. Foss-Feig, J. A. Gross, S. L. Rolston, and A. V. Gorshkov, "Optimal and secure measurement protocols for quantum sensor networks," *Physical Review A*, vol. 97, no. 4, p. 042337, 2018.

[5] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, "The security of practical quantum key distribution," *Reviews of modern physics*, vol. 81, no. 3, p. 1301, 2009.

[6] P. Komar, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin, "A quantum network of clocks," *Nature Physics*, vol. 10, no. 8, pp. 582–587, 2014.

[7] T.-Y. Chen, H. Liang, Y. Liu, W.-Q. Cai, L. Ju, W.-Y. Liu, J. Wang, H. Yin, K. Chen, Z.-B. Chen *et al.*, "Field test of a practical secure communication network with decoy-state quantum cryptography," *Optics express*, vol. 17, no. 8, pp. 6540–6549, 2009.

[8] M. Marcozzi and L. Mostarda, "Quantum consensus: an overview," *arXiv preprint arXiv:2101.04192*, 2021.

[9] R. G Sundaram, H. Gupta, and C. Ramakrishnan, "Efficient distribution of quantum circuits," in *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[10] R. G. Sundaram, H. Gupta, and C. Ramakrishnan, "Distribution of quantum circuits over general quantum networks," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2022, pp. 415–425.

[11] R. G. Sundaram and H. Gupta, "Distribution of quantum circuits using teleportations," in *IEEE International Conference on Quantum Software (QSW)*. IEEE, 2023.

[12] M. Hillery, H. Gupta, and C. Zhan, "Discrete outcome quantum sensor networks," *Physical Review A*, vol. 107, no. 1, p. 012435, 2023.

[13] C. Zhan, H. Gupta, and M. Hillery, "Optimizing initial state of detector sensors in quantum sensor networks," 2023, arXiv, 2306.17401.

[14] C. Zhan and H. Gupta, "Quantum sensor network algorithms for transmitter localization," in *IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2023.

[15] M. Ghaderibaneh, C. Zhan, H. Gupta, and C. R. Ramakrishnan, "Efficient quantum network communication using optimized entanglement swapping trees," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–20, 2022.

[16] M. Ghaderibaneh, H. Gupta, C. Ramakrishnan, and E. Luo, "Predistribution of entanglements in quantum networks," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2022, pp. 426–436.

[17] D. M. Greenberger, M. A. Horne, and A. Zeilinger, "Going beyond bell's theorem," in *Bell's theorem, quantum theory and conceptions of the universe*. Springer, 1989, pp. 69–72.

[18] D. Schlingemann and R. F. Werner, "Quantum error-correcting codes associated with graphs," *Phys. Rev. A*, vol. 65, p. 012308, Dec 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.65.012308

[19] M. Hillery, V. Bužek, and A. Berthiaume, "Quantum secret sharing," *Phys. Rev. A*, vol. 59, pp. 1829–1834, Mar 1999. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.59.1829

[20] M. Christandl and S. Wehner, "Quantum anonymous transmissions," in *Advances in Cryptology - ASIACRYPT 2005*, B. Roy, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 217–235.

[21] W. Dür, M. Skotiniotis, F. Fröwis, and B. Kraus, "Improved quantum metrology using quantum error correction," *Phys. Rev. Lett.*, vol. 112, p. 080801, Feb 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.112.080801

[22] M. Hillery, M. Ziman, V. Bužek, and M. Bieliková, "Towards quantum-based privacy and voting," *Physics Letters A*, vol. 349, no. 1, pp. 75–81, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0375960105014738

[23] P. Komar, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin, "A quantum network of clocks," *Nature Physics*, vol. 10, no. 8, pp. 582–587, 2014.

[24] E. T. Khabiboulline, J. Borregaard, K. De Greve, and M. D. Lukin, "Optical interferometry with quantum networks," *Phys. Rev. Lett.*, vol. 123, p. 070504, Aug 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.123.070504

[25] C. Meignant, D. Markham, and F. Grosshans, "Distributing graph states over arbitrary quantum networks," *Phys. Rev. A*, vol. 100, p. 052333, Nov 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.100.052333

[26] A. Pirker, J. Wallnöfer, and W. Dür, "Modular architectures for quantum networks," *New Journal of Physics*, vol. 20, no. 5, p. 053054, may 2018. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/aac2aa

[27] F. Hahn, A. Pappa, and J. Eisert, "Quantum network routing and local complementation," *npj Quantum Information*, vol. 5, no. 1, pp. 1–7, 2019.

[28] A. Fischer and D. Towsley, "Distributing graph states across quantum networks," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2021, pp. 324–333.

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

[30] G. Avis, F. Rozpędek, and S. Wehner, "Analysis of multipartite entanglement distribution using a central quantum-network node," *arXiv preprint arXiv:2203.05517*, 2022.

[31] J. Wallnöfer, M. Zwerger, C. Muschik, N. Sangouard, and W. Dür, "Two-dimensional quantum repeaters," *Phys. Rev. A*, vol. 94, p. 052307, Nov 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.94.052307

[32] S. de Bone, R. Ouyang, K. Goodenough, and D. Elkouss, "Protocols for creating and distilling multipartite ghz states with bell pairs," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–10, 2020.

[33] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *SIGCOMM*, 2020.

[34] K. Chakraborty, D. Elkouss, B. Rijsman, and S. Wehner, "Entanglement distribution in a quantum network: A multicommodity flow-based approach," *IEEE Transactions on Quantum Engineering*, 2020.

[35] M. Caleffi, "Optimal routing for quantum networks," *IEEE Access*, 2017.

[36] S. de Bone, R. Ouyang, K. Goodenough, and D. Elkouss, "Protocols for creating and distilling multipartite ghz states with bell pairs," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–10, 2020.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[38] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[39] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. Oliveira, S. Wehner *et al.*, "Netsquid, a discrete-event simulation platform for quantum networks," *Communications Physics*, 2021.

[40] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben *et al.*, "A link layer protocol for quantum networks," in *SIGCOMM*, 2019.

[41] B. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, 1988.

[42] P. van Loock, W. Alt, C. Becher, O. Benson, H. Boche, C. Deppe, J. Eschner, S. Höfling, D. Meschede, P. Michler, F. Schmidt, and H. Weinfurter, "Extending quantum links: Modules for fiber- and memory-based quantum repeaters," *Advanced Quantum Technologies*, vol. 3, no. 11, p. 1900141, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900141

[43] M. Steger, K. Saeedi, M. Thewalt, J. Morton, H. Riemann, N. Abrosimov, P. Becker, and H.-J. Pohl, "Quantum information storage for over 180 s using donor spins in a 28si "semiconductor vacuum"," *Science*, vol. 336, no. 6086, pp. 1280–1283, 2012.

[44] K. Saeedi, S. Simmons, J. Z. Salvail, P. Dluhy, H. Riemann, N. V. Abrosimov, P. Becker, H.-J. Pohl, J. J. Morton, and M. L. Thewalt, "Room-temperature quantum bit storage exceeding 39 minutes using ionized donors in silicon-28," *Science*, vol. 342, no. 6160, pp. 830–833, 2013.

[45] M. Zhong, M. P. Hedges, R. L. Ahlefeldt, J. G. Bartholomew, S. E. Beavan, S. M. Wittig, J. J. Longdell, and M. J. Sellars, "Optically addressable nuclear spins in a solid with a six-hour coherence time," *Nature*, vol. 517, no. 7533, pp. 177–180, 2015.

[46] P. Wang, C.-Y. Luan, M. Qiao, M. Um, J. Zhang, Y. Wang, X. Yuan, M. Gu, J. Zhang, and K. Kim, "Single ion qubit with estimated coherence time exceeding one hour," *Nature Communications*, vol. 12, no. 1, Jan 2021. [Online]. Available: http://dx.doi.org/10.1038/s41467-020-20330-w