Pre-Distribution of Entanglements in Quantum Networks

Mohammad Ghaderibaneh, Himanshu Gupta, CR Ramakrishnan, Ertai Luo
Stony Brook University, Stony Brook, NY, USA

***Abstract*—Quantum network communication is challenging, as the No-Cloning theorem in quantum regime makes many classical techniques inapplicable. For long-distance communication, the only viable approach is teleportation of quantum states, which requires a prior distribution of entangled pairs (EPs) of qubits. Establishment of EPs across remote nodes can incur significant latency due to the low probability of success of the underlying physical processes. To reduce EP generation latency, prior works have looked at selection of efficient entanglement-routing paths and simultaneous use of multiple such paths for EP generation.**

In this paper, we propose and investigate a complementary technique to reduce EP generation latency—to pre-distribute EPs over certain (pre-determined) pairs of network nodes; these pre-distributed EPs can then be used to generate EPs for the requested pairs, when needed, with lower generation latency. For such an pre-distribution approach to be most effective, we need to address an optimization problem of selection of node-pairs where the EPs should be pre-distributed to minimize the generation latency of expected EP requests, under a given cost constraint. In this paper, we appropriately formulate the above optimization problem and design two efficient algorithms, one of which is a greedy approach based on an approximation algorithm for a special case. Via extensive evaluations over the NetSquid simulator [1], we demonstrate the effectiveness of our approach and developed techniques; we show that our developed algorithms outperform a naive approach by up to an order of magnitude.

I. Introduction

Fundamental advances in physical sciences and engineering have led to the realization of working quantum computers (QCs) [2, 3]. However, there are significant limitations to the capacity of individual QC [4]. Quantum networks (QNs) enable the construction of large, robust, and more capable quantum computing platforms by connecting smaller QCs. Quantum networks [5] also enable various important applications [6–10]. However, quantum network communication is challenging—e.g., physical transmission of quantum states across nodes can incur irreparable communication errors, as classical procedures such as amplified signals or re-transmission cannot be applied due to quantum no-cloning [11, 12]. At the same time, certain aspects unique to the quantum regime, such as entangled states, enables novel mechanisms for communication. In particular, teleportation [13] transfers quantum states with just classical communication, but requires an a priori establishment of entangled pairs (EPs). This paper focusses on efficient generation of EPs in a quantum network.

Establishment of EPs over long distances is challenging. Coordinated entanglement swapping (e.g. DLCZ protocol [14]) using quantum repeaters can be used to establish long-distance entanglements from short-distance entanglements. However, due to low probability of success of the underlying physical processes (short-distance entanglements and swappings), EP generation can incur significant latency—of the order of 10s to 100s of seconds between nodes 100s of kms away [15–17]. Thus, we need to develop techniques that enable faster generation of long-distance EPs. In the past, researchers have addressed the above problem of high EP generation latency by developing techniques for selection of efficient entanglement routes [16, 18, 19] (or swapping trees [20]), and proposing to use multiple such routes/trees for each EP generation [18–20]. These approaches are effective, but can lead to heavy and bursty use of network resources at the time of EP request, and more importantly, the generation latency can still be high.

Pre-Distribution of EPs via Super-Links. In this paper, we propose a complementary *pre-distribution* approach to lower the EP generation latencies: in this pre-distribution approach, we proactively generate and pre-distribute/store EPs over certain (pre-determined) pairs of network nodes. When needed, these pre-distributed EPs can then be "used" to generate EPs across *requested* pair(s) of nodes; use of pre-distributed EPs, if carefully chosen, can result in lower generation latency for the requested EPs [21].[1] Note that the pre-distribution is complementary and can be used in conjunction with the prior approaches of efficient and multiple entanglement routes. The pairs of nodes that are chosen for pre-distribution of EPs are referred to as *super-links*, as they are tantamount to short-cuts in an entanglement-routing path. The above pre-distribution approach is particularly beneficial when we have a priori knowledge about the network traffic (e.g., the distribution of the EPs requests), which can be used to select an efficient set of super-links; in general, the selection of super-links should exploit the "commonality" across expected requests.

Contributions and Organization. For the above approach of pre-distribution of EPs to be most effective, we need to develop techniques to select an optimal set of super-links. In this context, our paper makes the following contributions.

- We motivate and formulate the optimization problem of super-link selection (SLS): Given a set of node pairs $\{(s, d)\}$ representing expected EPs requests, select a set S of super-links that results in minimum aggregate

[1] An extreme pro-active approach could be to continuously generate EPs at the same node pairs as the ones that will be requested (or if unknown, at all the node pairs in the network), but this can be very wasteful of network resources and in many cases, may not be useful or even viable.

generation latency of the $\{(s,d)\}$ EPs using S, under an appropriately defined cost constraint.

- For the above SLS problem, we design two algorithms: (i) Generalized Greedy (GG) Algorithm, a greedy approach based on an approximation algorithm for a special case of the SLS problem (§IV). (ii) Clustering Algorithm, based on the classical k-means clustering approach (§V).
- We develop a network protocol (§VI) for our proposed techniques, and generalize the developed techniques for certain variants of the SLS problem (§VII).
- Using extensive evaluations (§VIII), over the NetSquid simulator [1], we demonstrate the effectiveness of our approach and developed techniques; we show that our algorithms outperform the naive approaches by up to an order of magnitude.

To the best of our knowledge, no prior work has addressed the above optimization problem of selection of super-links for pre-distribution of EPs. The closest work is [21] which develops routing algorithms to leverage the pre-distributed entanglements in special network topologies, but does not address the problem of selection of super-links (referred to as *virtual links* in [21]). We start with presenting the relevant background in the following section (§II).

II. Background: Network Model, Entanglement Generation and Latency

In this section, we present the relevant background related to generation of EPs over remote nodes in a quantum network. We start with presenting our model of a quantum network.

Quantum Network. We consider a quantum network (QN) as 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. 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. Pairs of nodes connected by a link are defined as *adjacent* nodes. We follow the network model in [16, 20] closely. Fundamentally, each network node A should be capable of generating atom-photon EPs, so that it can generate an atom-atom EPs with an adjacent node B. *For clarity of presentation* and without loss of generality, we assume homogeneous network nodes with same parameter values. Each node is also equipped with a certain number of atomic memories to store the qubits of the atom-atom EPs. To facilitate atom-atom entanglement swapping (ES), each node is also equipped with an atomic-BSM device with an operation latency (t_b) and probability of success (p_b). Finally, there is an independent classical network with a transmission latency (t_c); we assume classical transmission always succeeds.

Quantum Communication via Teleportations and Entanglement Pairs (EPs). Direct transmission of quantum data is subject to unrecoverable errors, as classical procedures such as amplified signals or re-transmission cannot be applied due to quantum no-cloning [11, 12].[2] An alternative mechanism for quantum communication is *teleportation*, 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.

[2]Quantum error correction mechanisms [22, 23] 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 quantum networks.

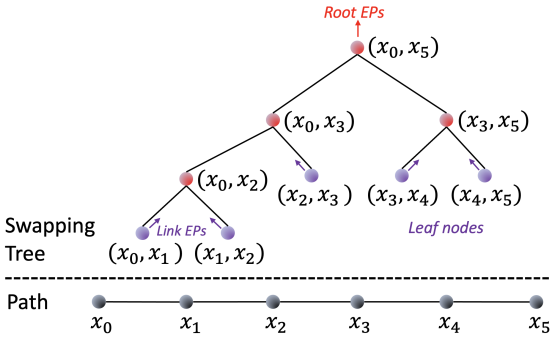

Fig. 1: A swapping tree over a path. The leaves of the tree are the path-links, which generate link-EPs continuously.

Generating EPs Across Remote Nodes. Establishing an EP over a pair (s, d) of remote nodes consists of two steps: (i) For some path $P = (s = x_0, x_1, x_2, \ldots, x_n, d = x_{n+1})$ from s to d in the QN graph with x_i's being the intermediate nodes, first we need to establish EP between every pair of nodes (x_i, x_{i+1}) for $0 \leq i \leq n$. (ii) Then, we conduct a series of entanglement swappings over these (x_i, x_{i+1}) EPs, to generate an EP over (s, d). The sequence in which the entanglement swappings are done has a bearing on the overall performance; this sequence of entanglement swappings is best characterized using a swapping tree [20] as defined below.

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) perform an ES over (A, B, C) to yield an EP over (A, C)—their parent. See Fig. 1. *Note that subtrees of a swapping tree execute in parallel.* Different swapping trees over the same path P can have different performance characteristics.

Estimating EP Generation Latency of a Swapping Tree. Since the entanglement swappings are stochastic, we may need to repeat the EP generation process—till it succeeds. Overall EP generation latency can be estimated as follows. 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. If T_{AB} and T_{BC} are exponentially distributed with the same generation latency of T, then we can derive [20] an expression for T_{AC} in terms of T.

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

Above, t_b and p_b are the operation latency and probability of success, respectively, of the atomic-BSM device at B, and t_c is the transmission latency of the classical bits. The above makes

an assumption of $T_{AB} = T_{BC}$, which holds in "throttled" trees that uses minimal underlying network resources [20]. To complete our estimation of generation latency, we also need to estimate EP generation latencies at the leaves of the swapping tree (i.e., network links). Link EP generation latency can be estimated using node and link parameters, as derived in [20]; we omit the tedious details, as they are not important or very relevant to our work. Finally, we note that above we have implicitly assumed a `Waiting` protocol of generation EPs wherein a qubit of an EP waits (being stored in a quantum memory) for its counterpart to become available so that an ES operation can be performed. In the alternate protocol, viz., `WaitLess`, all the links EPs are synchronized and BSMs conducted simultaneously—obviating the need for memories with higher decoherence times. We assume the `Waiting` protocol, as it is a more efficient approach for generating remote EPs. However, our techniques are also applicable to the `WaitLess` model.

III. Super-Links Selection (SLS) Problem

In this section, We motivate and formulate the problem of selection of super-links addressed in our paper.

Pre-distribution of EPs. Consider a set of source-destination pairs $\{(s, d)\}$ that require an EP shared across them, from time to time. The EP across an (s, d) pair may be required to teleport a qubit state from s to d, as an example. Without any pre-existing EPs, distributing an EP over (s, d) can incur very high latency due to stochastic underlying processes with low-probability success; in fact, the latencies can be of the order of 10s of seconds [16, 17]. One approach to reduce the EP generation latency is to use multiple independent swapping-trees simultaneously—this approach has been explored in recent works [18–20]. In this paper, we propose a rather *proactive* approach of latency reduction, wherein we proactively generate and distribute EPs at appropriately selected pairs of nodes (not necessarily, the original source-destination pairs).[3] Such pre-distribution of EPs is done in a way so that the latency incurred in a real-time EP request is minimized. In the extreme case, we could pre-distribute EPs directly over the expected (s, d) pairs, which would result in zero latency of teleportation requests, but such a strategy would be wasteful of network resources. Thus, in this paper, we address an optimization problem of determining an efficient pre-distribution strategy. We start with a definition below.

Super-Links (SL). Consider a quantum network, and a pair of connected nodes A and B. Let us decide to continuously generate and maintain EPs over (A, B), to aid in future EP requests across certain network node pairs. We call (A, B) a *super-link (SL)*, since the pair (A, B) can essentially be now looked upon as a "link" that is generating EPs continuously generated over it.[4] **Each super-link (A, B) is implicitly associated with a path** P_{AB} and a swapping-tree over P_{AB} which is used to generate the EPs over (A, B); in this work, we determine the optimal path and swapping-tree over it using the schemes in our recent work [20]. The main purpose of super-links is to help reduce the latency of real-time EP requests over other network nodes.

[3]This proactive approach of pre-distributing EPs is complementary to the multiple paths/trees approach, and can be used together with it (see §IX).

[4]Prior work [21] has used the term virtual links, instead.

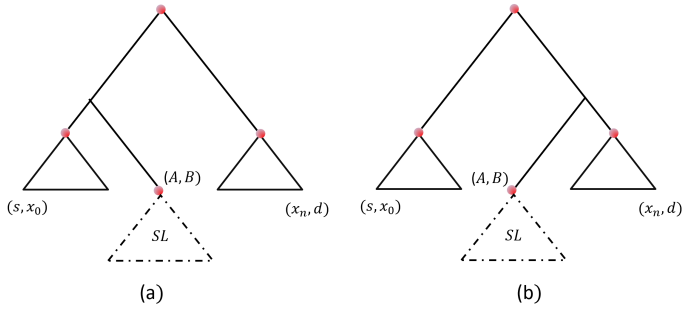

Fig. 2: Two possible swapping trees to generate EP over (s, d) pair using a super-link (A, B).

To illustrate the usefulness of an SL, consider a source-destination pair (s, d) of network nodes and let T be the expected latency incurred in generating an EP over (s, d) without any SL. Consider the super-link (A, B), and assume that there are multiple EPs available over (A, B) at any point of time. Then, an EP over (s, d) can possibly be generated with latency less than T by using the EPs over (A, B), depending on the paths (s, A), and (d, B). In particular, we can first establish EPs over (s, A) and (B, d) using appropriate swapping trees, and then use a separate swapping tree to generate an (s, d) EP using EPs from (s, A), (A, B), and (B, d). See Fig. 2. Below, we derive the latency incurred in such an SL-based strategy. Note that this overall "path" (s, A, B, d) may be very different than the shortest/optimal path between s and d to generate an (s, d) EP.

EP Generation Latency using a SL. We can use a modified Eqn. (1) to compute the expected latency of the swapping trees shown in Fig. 2 which use EPs already available at SL (A, B) to generate an EP at (s, d). If $T_{s,A}$ is the latency to generate a successful EP over (s, A) (using some swapping-tree) and p_b is the atomic-BSM probability of success, then the expected latency to generate a successful EP over (s, B) is $T_{s,B} = (T_{s,A} + t_b + t_c)/p_b$ if we assume sufficiently many EPs available over (A, B). Then, the expected latency to generate an EP over (s, d) using the swapping tree in Fig. 2(a) is

$$T_{sd} = (\frac{3}{2}\max((T_{s,A} + t_b + t_c)/p_b, T_{B,d}) + t_b + t_c)/p_b \quad (2)$$

based on a slight modification of Eqn. (1). Similarly, the expected latency to generate an EP over (s, d) using the swapping tree depicted in Fig. 2(b) can be given by:

$$T_{sd} = (\frac{3}{2}\max(T_{s,A}, (T_{B,d} + t_b + t_c)/p_b) + t_b + t_c)/p_b \quad (3)$$

The overall effective expected latency to generate an (s, d) EP is the minimum of the above two quantities, as the better of the two swapping-trees in Fig. 2 would be chosen.

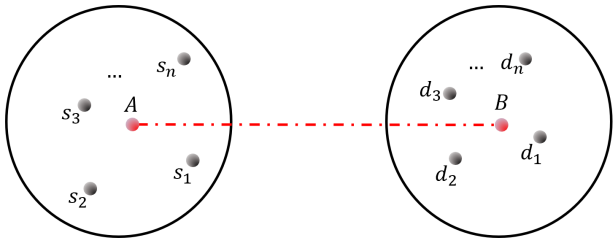

Fig. 3: A single super-link (A, B) can be very effective in handling *all* EP requests of the type (s_i, d_j).

Number of EPs Required at Each SL. Above we assumed that sufficiently many EPs are available at an SL. However, it is easy to see that the expected number of EPs used to generate an (s, d) EP is $1/(p_b)^2$ — this holds for either of the swapping trees in figure. This is true since the expected number failures that an (A, B) EP may be involved in is ${p_b}^2$ as the height of either tree is 2.

Motivation for Selection of SLs. The above discussion illustrates how a super-link can be useful in reducing latency. However, an SL-based EP generation scheme can also be wasteful of resources, depending upon the distribution of future requests. In general, the SL-based strategy can be effective if the EPs maintained at any particular SL can be useful in generation of many (s, d) EPs. For example, consider a set/cluster of sources S and a cluster of destination D. Let us say that, periodically, we need to serve an EP request over some (s_i, d_i) pair where $s_i \in S$ and $d_j \in D$. For such an EP demand model, just a single super-link (A, B) could be very useful wherein A is close to all the nodes in S and B is close to all the nodes in D. Essentially, an EP over (A, B) can be used to efficiently generate an EP over any (s_i, d_j) pair where $s_i \in S$ and $d_j \in D$. See Fig. 3. In general, for a given EPs demand distribution, there is a need to determine a efficient set of SLs that will be most effective in serving the given demand distribution. We formulate this optimization problem in the next subsection.

A. SLS Problem Formulation

In this subsection, we formalize the *selection of super-links (*SLS*)* optimization problem, which is to efficiently select super-links (SLs) in a given network EPs to effectively handle a given demand distribution of EPs in the network. We start with describing the overall setting.

Problem Setting. Consider a quantum network and a set of source-destination pairs $S = \{(s, d)\}$. Let us assume the time to be divided in slots of size τ, and that at the start of each time slot, there is a single EP request for one of the (s, d) pairs in S.[5] In order to serve these requests with minimum latency, we select a set of SLs based on the below formulated optimization problem, and continuously generate EPs over these SLs. When an (s, d) EP request is received, an EP over (s, d) is generated using an efficient swapping-tree using already-available EPs at one of the SLs (say l). Also, while the request is being served, we stop generating EPs over any SL other than l whose associated path intersects with the (s, d)'s EP-generation path (to allow for most efficient generation of EP over (s, d)). Throughout the paper, we implicitly assume that we use at most one SL in generating an EP for an (s, d) pair. The main motivation behind this assumption to limit the number of EPs required at any SL (e.g., if we use two SLs in an (s, d) swapping tree, then the number of EPs required would be $1/{p_b}^4$). Before formulating the optimization problem addressed here, we define two functions.

Objective Function ψ. Consider a set S of (s, d) pairs and a set of super-links L. We define a function $\psi(S, L)$ as the average generation latency of the pairs in S using the super-links in L. In particular, let $T((s, d), l)$ denote the expected latency in generating an EP pair over (s, d) using a super-link l; this is essentially a minimum of the two qualities given in Eqns. (2)-(3) where $l = (A, B)$. Then, the function $\psi(S, L)$ is essentially given by:

$$\psi(S, L) = \sum_{(s,d) \in S} \min_{l \in L} T((s, d), l).$$

Cost Function: $Cost(L)$. To capture efficient utilization of network resources, we now model the cost of super-links. There are many ways to model the cost—but in general, the idea is for the cost to reflect the network resources consumed in generation of EPs over the set of SLs. One simple way to measure the network resources consumed in generating an EP could be just the latency incurred in generating it—and based on this, the cost of an SL could be defined as the expected generation latency of an EP over the SL using the associated swapping-tree. In our discussion, however, we capture the network resources consumed more accurately by modelling it in terms of the aggregate number of EP generation attempts made by the links in the path (which are also the leaves of the associated swapping-tree) associated with the super-link, in generating an EP at the SL. In other words, the cost of an SL l is the number of BSM failures incurred at the link-level of the swapping tree associated with the SL l, in generation of an EP at the SL.[6] Thus, the cost of a set L is just the sum of the costs of the individual super-links. We note that the techniques developed in this work is independent of the cost model for super-links.

Selection of Super-Links (SLS**) Problem.** Given a quantum network and a set S of (s, d) pairs, the SLS problem is to select a set L of super-links that minimizes $\psi(S, L)$ such the total cost of the super-links L is at most a given budget.

Hardness and Variants. The SLS problem can be easily shown to be NP-complete, from a simple reduction from the well-known set cover problem. One could look at other variants of the problem – e.g., minimize the maximum latency of the

[5]The derivation of Eqns. (2) and (3) implicitly assume that τ is large enough that the EPs for all the SLs are available when a new (s, d) EP request arrives, i.e., τ is larger than the maximum latency incurred in generating the desired number of EPs over an SL in the network. Note that EPs across SLs are generated independently and in parallel, as they use disjoint paths for EP generation.

[6]In reality, we generate multiple EPs at each SL, but since this number is uniform is across all SLs, the cost can be based on generation of a single EP.

(s, d) pairs, minimize the cost of the SLs under the constraint that the latency of each (s, d) pair is bounded by a given constant, etc. The algorithms designed in our paper can be modified to address these variants.

B. Related Works.

There have been a few works in the recent years that have addressed generating long-distance EPs efficiently. These works have focused on selecting an efficient routing path [16, 18, 19] or swapping-tree [20] for the swapping process. The closest work in the quantum communication literature to ours is [21], which shows that use of pre-distributed entanglements can reduce the latency of generating entanglements, and develops routing algorithms in special network topologies (e.g., rings, grids) that leverage these pre-distributed entanglements; we note that [21] refers to the our super-links as virtual links. Our work focuses on the related problem of selection of links where these pre-distributed entanglements should be generated—given a distribution of future entanglement requests. In addition to the above works, other works that have focused on developing quantum networking protocols include [24] which develops a link-layer protocol for entanglement generations, [25] which develops a network protocol for efficient use of entanglement-pairs for swapping operations in entanglement generation over remote network nodes.

Our proposed optimization problem SLS of selection of super-links has not been studied before. The closed optimization problem studied before is that of *adding short-cuts in a graph (*APSG*)* by Myerson [26]. The APSG problem addressed in [26] proposes to add a certain number of short-cuts to a given graph with the objective of minimizing the average path length across all pairs of nodes. They propose approximation algorithms for the above graph problem based on known approximation algorithm for the k-medians optimization problem. The short-cuts in our SLS problem can be looked upon as short cuts in the APSG problem. For the APSG problem, [26] consider both versions: allowing multiple short-cuts in a path, or only a single short-cut for each path. However, the fundamental differences of our SLS problem with the above APSG problems are: (i) In APSG, the short-cuts are assumed to be of uniform weight (they use a certain *number* of short-cuts; generalization of their techniques to weighted short-cuts requires fundamentally new techniques, because the weighted version of k-medians problem itself has not been studied before to the best of our knowledge. (ii) The short-cuts in APSG are disjoint and thus independent by definition, while in our SLS problem, the biggest challenge arises from the inter-dependence been the super-links (see §IV. In addition to the above, other related work includes [27] which addresses a problem related to the dual version of the SLS problem; in particular, [27] look at *mincost distance-d* problem which attempts to bound the distance between all pairs of nodes to d by adding short-cuts of total minimum cost. Here, the cost of the short-cuts is non-uniform. The *mincost distance-d* is related to the dual of the SLS problem, wherein we want to add super-links of total minimum cost such that the EP generation latency of each (s, d)-pair is bounded by a certain constant; however, [27]'s notion of distance between pairs of nodes is much simpler compared to the notion of EP generation latency, and the techniques in [27] do not generalize to the SLS-dual problem (and certainly, not to the SLS problem addressed here).

IV. Greedy Algorithm

In this section, we design a greedy algorithm for selection of SLs. At a high-level, a greedy approach for the SLS problem would iteratively select an SL based on some criteria, until the given budget is exhausted. For optimization problems with a submodular objective function, an appropriately designed greedy approach can be shown to deliver a solution whose objective value is within a constant-factor of the optimal objective value, i.e., a constant-factor approximate solution. Unfortunately, the SLS problem's objective function is not submodular, and hence, a greedy approximation algorithm is not feasible. Nevertheless, an appropriately designed greedy algorithm can be expected to deliver good solutions in practice. Below, we start by motivating designing a greedy approach for the SLS problem, by showing that for a special case of the SLS problem, a simple greedy algorithm delivers a constant-factor approximate solution. With this motivation and insight, we design an appropriate greedy algorithm for the general SLS problem; the designed greedy algorithm delivers good solutions in practice as shown in our empirical results (§VIII).

GD−SLS **Special Case: Given Disjoint SL Candidates.** Consider a special case of the SLS problem, wherein we are *given* a set L of *disjoint* paths in the network and the problem is to select a subset $L' \subseteq L$ of SLs that yields the minimum aggregate latency of the (s, d) pairs under the cost constraint. We refer to this special case as the GD−SLS problem. For this special case, a straightforward greedy approach of iteratively selecting the SL that has the maximum "benefit" per unit cost can be shown to deliver a constant-factor approximate solution. The benefit of an SL l is defined as the reduction in the aggregate latency of (s, d) pairs due to adding l as an SL; we formalize this notion of benefit below.

Benefit of an SL. Consider a stage of the greedy algorithm where a set of SLs L' have already been selected. At this stage, the *benefit* of an SL $l \notin L'$ is defined as

$$\psi(L') - \psi(L' \cup \{l\}),$$

where the ψ function, as defined earlier, is the aggregate latency of the given (s, d) pairs using the set of SLs.

We now give the pseudo-code of the proposed greedy approach for the GD−SLS problem.

The aggregate latency function can be easily shown to be monotonic and submodular, in the context of the special-case of GD−SLS. Thus, the Greedy algorithm can be shown to deliver a solution with a near-optimal benefit.

Theorem 1: For the special case GD−SLS problem, the Greedy algorithm delivers a solution that has a benefit of at least 63% of the benefit of the optimal solution. ■

Algorithm 1 Greedy for GD−SLS Problem.

Input: Quantum Network, $\{(s,d)\}$ pairs, L of candidate SLs, Cost Budget C.
Output: Set L' of Super-Links.
$L' = \phi$
while $(\text{cost}(L') < C)$ and $(L - L' \neq \phi)$ **do**
    Let l be the SL in $L - L'$ with the highest value of
$$(\psi(L') - \psi(L' \cup l))/Cost(l)$$
    $L' = L' \cup \{l\}$
Return L'

Generalized Greedy (GG) for the SLS Problem. Note in the general SLS problem, there are no given candidates; in fact, each path in the network can be an SL – so, the number of candidate SLs is exponential in the number of network nodes. However, it is reasonable to consider only the best (i.e., ones with lowest-latency swapping trees) path for every pair of nodes in the network; this yields at most $O(n^2)$ candidate SLs, where n is the number of nodes in the network. Let us use $\mathcal{P}$ to denote this set of candidate SLs. The set of candidate SLs in $\mathcal{P}$ might however be non-disjoint, i.e., have a common node or link. Intersecting candidate SLs creates multiple issues: (i) The objective function is not submodular for SLs with intersections; (ii) Allowing non-disjoint SLs requires appropriate allocation of node and link resources across SLs to generate EP for the SLs, and more importantly, would results in higher generation latencies for the EP over the SLs. For the above reasons and also for simplicity, we enforce the condition that the selected SLs be disjoint. With that condition in mind, a straightforward greedy approach for the general SLS problem could be to iteratively select the SL (from the candidate set $\mathcal{P}$) with the highest benefit per unit cost among those that do not intersect with the already selected SLs. This simple approach can easily perform badly (see Fig. 4), but can be further improved in one of the following options: (i) Allow for non-shortest paths not in $\mathcal{P}$, if needed; (ii) Allow removal of previously selected SLs if they intersect with a promising candidate later. In our greedy algorithm for the general SLS problem, which we refer to as GG, we incorporate both the above options. We present GG's pseudo-code as Algorithm 2 below. In GG, we essentially evaluate each pair of nodes (u,v) as a potential SL for addition to the set of SLs S being maintained. We consider two options as follows (see Fig. 4): (i) *Update:* If we allow deletions from S, then we pick the shortest path P_U from u to v in the original graph, delete all the SLs from S whose associated paths intersect with P_U. (ii) *Append:* If we do not allow deletions from S, then we instead pick the shortest path P_A from u to v in the "residtual" graph to avoid intersections with associated paths of SLs in S. For each potential option (two for each pair of nodes), we compute the ratio of increase in benefit to increase in cost, and pick the option with the best ratio. We iterative the above, until the given cost budget is exhausted.

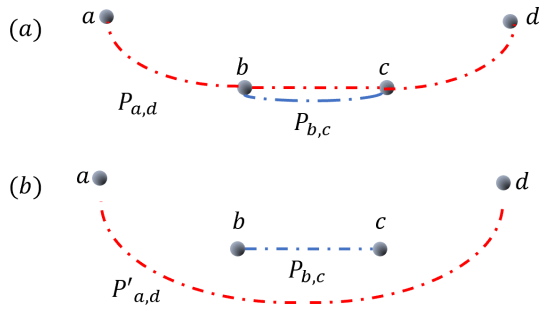

Fig. 4: Illustrating the simple greedy scheme, and the GG Algorithm with *Update* and *Append* options. (a) Two SLs (a,d) and (b,c) shown with associated (shortest) paths P_{ad} and P_{bc} respectively; these paths are not disjoint. (b) New non-shortest path P'_{ad} for (a,d) which doesn't intersect with (b,c)'s associated path P_{bc}. If SL (b,c) (with associated path P_{bc}) is first/already chosen as an SL in the solution, then a *simple greedy algorithm* will not consider (a,d) as a super-link since its associated path P_{ad} intersection with P_{bc}. Instead, the GG Algorithm considers including super-link (a,d) in the solution via two options: (i) *Update:* Remove (b,c), and add (a,d) with associated (shortest) path P_{ad}, or (ii) *Append:* Add (a,d) with associated path P'_{ad}.

Algorithm 2 Generalized Greedy (GG) Algorithm for SLS

Input: Quantum Network G, $\{(s,d)\}$ pairs, Cost Budget C
Output: Set L of Super-Links.[7,8]
$L = \phi$
while $\text{cost}(L) < C$) **do**
    for $(u,v) \in ((V \times V) - L)$ **do**
        /* *Update*: If added with deletions from L */
        P_U = shortest path between (u,v) in G
        X = set of SLs in L whose associated paths are disjoint with P_U.
        $L'_U = X \cup \{(u,v)\}$
        $b_U = \big(\psi(L) - \psi(L'_U)\big)/(Cost(L'_U) - Cost(L)$ [9]
        /* *Append*: If added without any deletions from L */
        P_A = shortest path between (u,v) in the "residual" graph G' where G' is G without the nodes and edges used in the paths associated with SLs in L).
        $L'_A = L \cup \{(u,v)\}$
        $b_A = \big(\psi(L) - \psi(L'_A)\big)/Cost(\{(u,v)\}))$

        if $b_A > b_U$ **then**
            $b_{u,v} = b_A$
            $L'_{u,v} = L'_A$
        else
            $b_{u,v} = b_U$
            $L'_{u,v} = L'_U$
    $L' \leftarrow L'_{u,v}$ with maximum $b_{u,v}$ value.
    if $L = L'$ **then**
        break
    $L = L'$
Return L

V. Clustering-Based Algorithm

In this section, we design an algorithm based on clustering the (s, d) pairs and picking an SL for each of the clusters.

Intuition and Basic Idea. Intuitively, the idea is to pick SLs each of which can effectively serve as many (s, d) pairs as possible, to allow for efficient use of resources. Therefore, given a set of (s, d) pairs, we want to partition the set into a small number of clusters in a way that each cluster of (s, d) pairs can be effectively served by an appropriately picked single SL. We can use a clustering algorithm based on the standard k-means clustering algorithm. For a given k, the k-means algorithm starts with a randomly chosen k cluster-centroids (SLs in our case) and iteratively (i) assigns each (s, d) pair to the "closest" centroid SL; the set of (s, d) pairs assigned to the same centroid SL forms a cluster; (ii) computes new centroid SLs for the clusters formed; (iii) repeat until some condition is satisfied. Since in our SLS problem, the constraint is in terms of the total cost budget rather than the number of clusters k, we iterate over potential k values and pick the best clustering solution that satisfies the budget constraints. Even for a given k, to use the above k-means approach, we still need to define/determine: (a) The distance between an SL and an (s, d) pair to determine the closest SL for centroid assignment, and (b) How to compute a centroid for a given set of (s, d) pairs. We define these aspects as follows.

- *Distance between SL and an (s, d) pair.* The distance between an (s, d) pair and a SL can be defined as the latency of the (s, d) pair using the SL. Thus, all the (s, d) pairs in the same cluster would have the common property that the centroid SL of this cluster reduces their latency the most compared to other centroid SLs.
- *Centroid SL of a set of (s, d) Pairs.* We use an exhaustive way to find the updated centroids for each cluster. In particular, for each cluster C_i, we assume any of the paths p in the network can be the new centroid, and traverse through all the paths in the network and pick the one whose aggregate distance (as defined above) from the (s, d) pairs in the cluster is minimum.

We now present the overall pseudo code below.

Algorithm 3 Clustering Algorithm for SLS Problem

Input: Quantum Network G, S = $\{(s, d)\}$, Cost Budget C.
Output:
for $k = 1, 2, \ldots, |S|$ **do**
    $L_k = \text{ClusteringViaKMeans}(G, S, k)$
Return L_k where $k = \arg\min_{k=1,2,..} \psi(L_k)$

[7]As mentioned before, although a super-link is defined by a pair of nodes, it is also associated with a path used for the swapping-tree.

[8]A super-link is considered only if its associated path is able to generate $1/(p_b)^2$ EPs with an expected latency of less than τ.

[9]The special case is when the denominator is negative while the nominator is positive; such an SL is considered better than those with positive b values.

Algorithm 4 Selecting k Super-Links via k-means Clustering

Function Name: ClusteringViaKMeans
Input: k, Quantum Network G, $S = \{(s, d)\}$ pairs, Cost Budget C.
Output: Set L of Super-Links (also, centroids of the k clusters).
Randomly pick k SLs/centroids $l_1, l_2, \ldots, l_k$.
while () **do**
    /* Assignment Stage */
    for each pair $(s, d) \in S$ **do**
        Find the SL l_j with minimum latency $\psi((s, d), l_j)$
        Assign (s, d) to the SL/centroid l_j.
    /* Update SLs/Centroids */
    for $1 \leq i \leq k$ **do**
        Let S_i be the set of (s, d) pairs assigned to SL l_i.
        Let l be the super-link[9] (with an associated path and swapping-tree) with minimum $\psi(S_i, l)$, .
        $l_i = l$
    /* Determine whether to break. */
    Keep track of the set of SLs with best $\psi(S, \{l_1, l_2, \ldots, l_k\})$, and **break** if the best set of SLs doesn't change in 5 iterations of the while loop.
if $Cost(L) \geq C$ then reduce $Cost(L)$ by "shortening" (i.e., using a sub-path) some $l \in L$
Return $\{l_1, l_2, \ldots, l_k\}$

VI. Network Protocol and Implementation

In this section, we present the network protocol with other implementation details, used to generate EPs over selected SLs and then, to use these pre-distributed EPs over SLs to serve incoming communication requests. Overall, the network protocol has the following high-level components/steps:

- **Selection of SLs.** This step is largely done offline at a centralized (classical) node, with the inputs being the EP requests (i.e., the weighted set of (s, d) pairs) and the quantum network details. However, as the request distribution changes, the set of SLs are updated.
- **Continuously Generate EPs across Selected SLs.** We continuously generate EPs over the selected SLs, using all the link resources available (note that our selected SLs are disjoint), so as to keep them as "fresh" as possible. As we use at most one SL per swapping-tree of a future (s, d) EP request, we store $1/{p_b}^2$ EPs at each SL since that is the expected number of EPs needed at an SL. In addition, we also keep track of the generation time of the qubits in each EP pair, so that we can appropriately determine if they can be used. Generation of EPs across the SLs is done using the swapping-tree protocol described below.
- **Serve Communication Requests.** Note that for any (s, d) EP request, we already know the SL that we will use in its swapping-tree—since, such assignment of SLs to (s, d) pairs is implicitly already done during selection

of SLs.[10] For a given (s, d) pair request, let (x, y) be the SL used. Now, consider the sequence of nodes (i.e., the entanglement path) $\langle s, \ldots, x, y, \ldots, d\rangle$, over which the swapping-tree is constructed. We discuss generation of EP across (s, d) using such a swapping-tree involving the SL (x, y) below. Note that we need to only generate one EP (see §VII for the general case).

EP Generation over a Given Swapping Tree. Consider a a swapping tree T for a path P connecting a pair of nodes (x, y) in the network. We build our protocol on top of the link-layer of [24], which is delegated with the task of generating EPs on a link at a desired rate. In our case, we select disjoint SLs—thus, each link of on SL is involved in a single SL swapping-tree;[11] Thus, each link in an SL can continuously generate EPs at the maximum rate possible, unless instructed otherwise (see below). As mentioned before, for a given SL (and a corresponding path), we compute the optimal swapping-tree using techniques in [20]. The network protocol that generates EPs over a *given* swapping tree can be described as follows. For simplicity, let us first consider generation of a *single* EP at a time; we consider multiple EPs shortly. The key aspect of the protocol is that the entanglement-swapping (ES) operation is done only when both the appropriate EP pairs have arrived. The key step of the protocol is the bell-state measurement (BSM) or the ES operation done at appropriate nodes on the path. Consider a pair of EPs over (A, B) and (B, C), with the node B now planning to do an ES operation to generate an EP over (A, C); this situation corresponds to siblings (A, B) and (B, C) and their parent (A, C) in the swapping tree. This BSM operation at B essentially entails the following:

- If the swapping (BSM) operation at node B succeeds, it transmits classical bits to C which manipulates its qubit, and send the final ack to the other end-node A to finalize "generation" of the EP over (A, C).
- If the swapping (BSM) operation fails, a classical `ACK` is send to the end nodes A and C to discard the corresponding qubits.

Generating Multiple EPs. The above discussion is to generate a single EP at a time. To generate multiple EPs, we can do the above simultaneously–not necessarily independently. Essentially, each node in the swapping tree (including the link leaves) stores the EPs that are valid, i.e., that are part of an EP and haven't decohered. The node B (as define above) will initiate an ES operation as soon as (A, C) as well as (C, B) has at least one EP each. Note that we can even use different swapping-trees for multiple EPs, even over the same path.

Serving Communication Requests. Consider the sequence of nodes (i.e., the entanglement path) $\langle s, \ldots, x, y, \ldots, d\rangle$, over which the swapping-tree is constructed; here (x, y) is the SL being used to generate an EP over (s, d). Now, to generate an EP over (s, d), we essentially use the above protocol as detailed above for a swapping-tree, except for the following: (i) Treat (x, y) as a link with EPs available, (ii) The link (x, y) continues generating EPs using its swapping tree, independent of the swapping tree used for $\langle s, \ldots, x, y, \ldots, d\rangle$. (iii) For best performance, during EP generation of (s, d), we do not use any link on the path (s, x) or (s, y) for generation of any other EPs (i.e., EPs from other selected SLs); other than this constraint, other SLs continue to generate the EPs over them. See the example in Fig. 5, where with the help of a (x_2, x_4) EP generated from a (x_2, x_4) super-link, a smaller swapping-tree can be used to generate the (x_0, x_6) EP.

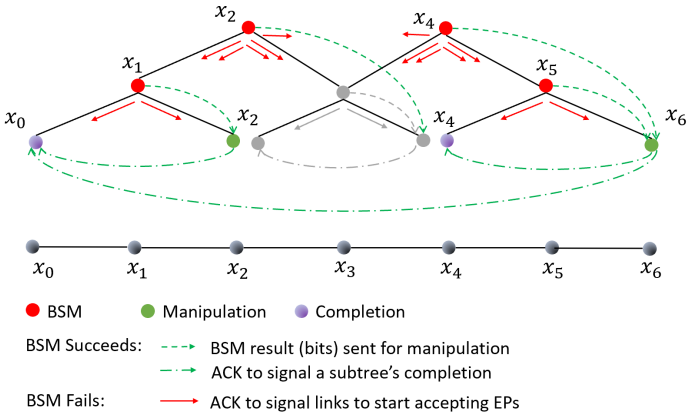

Fig. 5: Serving a communication request over (x_0, x_6), and the ACKs generated following a BSM success or failure.

VII. Generalizations and Discussion

In this section, we relax some of the simplifying assumptions made earlier, discuss other variants of the `SLS` optimization problem, and other issues.

General Distribution of EPs Requests. Throughout the paper, we have considered a simple model of EP requests wherein there is a single (s, d) EP request in each time slot. Our techniques can be easily generalized to handle more general request models: (i) A weighted model, wherein a weight (a fraction) associated with an (s, d) pair signifies the ratio of times a request belongs to this (s, d) pair. For this weighted model, the benefit function can be easily extended by using a weighted benefit from each (s, d) pair, and thus, the greedy algorithm and its variants can be easily extended. Similarly, the clustering approach also generalized by appropriately extending the centroid of a set of pairs to be based on weighted distance to the (s, d) pairs in the cluster. (ii) We can also consider the generalization to multiple (s, d) requests in a timeslot. There are multiples ways to handle this generalization. We could storing more EPs with each SL, and/or have multiple SLs assigned to each (s, d) pair so that at the time of request one SL is serving exactly one (s, d) pair.

Low SL Replenishment or Decoherence Times. We had earlier assumed that the time to replenish the SLs, i.e., the latency incurred to generate the desired EPs over all the

[10] In a more sophisticated scheme, we could determine the SL to be used by an (s, d) pair at real-time based on the current ages of the EPs available at the SLs. We will explore this generalization in our future work.

[11] If a link (a, b) is part of multiple swapping-trees of different SLs, it needs to handle multiple link-layer requests at the same time. Such link-layer requests can be implemented by creating "virtual" independent atom-photon generators at a and b, with one pair of synchronized generators for each link-layer request. See [20] for more details.

selected SLs (this happens in parallel across the SLs, as they are disjoint), is less than the time slot τ. If that is not the case, then, as expected, the benefit of SLs is reduced but can be still significant. In essence, the impact of high replenishment time is that the EPs at SLs may not be readily available when an (s, d) request comes and hence the latency reduction in EP generation over (s, d) s reduced. Note that EPs over SLs continue to be generated as (s, d) EP is being generated.

Similarly, lower memory decoherence time reduces the impact of our overall scheme of pre-distributing EPs, but can be minimized by somewhat synchronizing the generation of SLs EPs with that of the (s, d) requests so that – the SL EPs are as "fresh" as possible when needed by the (s, d) request's swapping tree. Note that we do not select SLs whose expected latency of generating EPs is more than the decoherence time.

Other Optimization Objectives. In this paper, we are focused on the optimization objective of minimizing the aggregate latency of given set of (s, d) pairs under the budget constraint of total cost of super-links selected. However, some related optimization objectives can be equally important. E.g., one can consider the optimization objective of maximizing the number of (s, d) pairs whose latency can be lowered to a certain bound d using SLs, under the given budget constraints. This optimization problem can be looked upon as a coverage problem, where an SL covers an (s, d) pair if it helps reduce the latency below the given bound, and appropriate techniques similar to ours can be designed. Another variant is to minimize the maximum latency of an (s, d) pair, under given budget constraints; this variant can be solved by an iterative application of the previous variant, by trying decreasing d values. Finally, one can look at the dual variants of the `SLS` problem, wherein the objective could be to minimize the lost of the SLs selected, while ensuring some bounds/constraints on (s, d) latencies.

VIII. Evaluations

In this section, we evaluate the performance of our protocols/algorithms in terms of the generation latency of the requesting EPs, by implementing and evaluating them on top of the discrete event simulator for QNs called NetSquid (Python/C++) [1]. The NetSquid simulator accurately models various QN components/aspects, and in particular, we are able to (i) define various QN components, viz., nodes, quantum memory, quantum channels, etc., and (ii) simulate swapping-trees based entanglement generation by implementing gate operations needed in entanglement swapping or teleportation. Our algorithms are implemented in Python.

Simulation Setting. We generate random quantum networks in a similar way as in the recent works [18, 20]. By default, we use a network spread over an area of $100km \times 100km$. We use the Waxman model [28], 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).

EP Generation Parameter Values. We use EP generation parameter values as the ones used in [16, 20], and vary some of them. In particular, we use atomic-BSM probability of success (p_b) to be 0.4 and latency (t_b) to be 10 μ secs. The optimal-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 μsec and 0.33 respectively. Finally, we use photon transmission success probability as $e^{-d/(2L)}$ [16] where L is the channel attenuation length (chosen as 20km for an optical fiber) and d is the distance between the nodes. We vary the cost budget (see §III) used by our algorithms—from 5000 to $40,000$ units with the default value of $20,000$.

EPs Request-Traffic Model. We pick the (s, d) pairs such that the distance with s and d is within a range 30 to 120 kms. We vary the number of (s, d) pairs from 4 to 20 with the default value of 12. As described in §III-A, we divide the simulation duration into slots of 4 seconds each, and at the start of each slot, there is a request to establish an EP over one (randomly chosen) of the (s, d) pairs.

Algorithms and Performance Metrics. We use the EP generation algorithm from the latest work [20] as a benchmark for EP generation without super-links; the algorithm from [20] is based on an dynamic programming algorithm that determines an *optimal* swapping-tree over a given pair of nodes. We refer to this non-superlink algorithm as `Non-SLs` in the plots. For selection of super-links and EP generation over (s, d) pair with super-links, we implement and compare the following algorithms: `Naive`, Generalized Greedy (`GG`; §IV), and Clustering Approach (`CLUS`; §V). The `Naive` algorithm is essentially the same as our `GG` algorithm, except that the only candidate super-links considered are the sub-paths of the shortest path for a given (s, d) pair. In addition, we also compare certain variants of the `GG` algorithm in one of the experiments. We compare the above algorithms mostly in terms of the average generation latency in seconds of an EP over an (s, d) pair. In one of the plots, we also evaluate the total latency of the EPs generated over the super-links.

Evaluation Results. We start with comparing the performance of the various `SLS` algorithms in terms of the average EP generation latency time to establish an EP over an (s, d) pair. See Fig. 6, where we plot the average generation latency for various algorithms for varying cost budget (over the cost of the super-links selected), network edge density, number of network nodes, and number (s, d) pairs. We observe that all algorithms result in lowered average EP generation latency, with our main algorithm `GG` significantly outperforming all the other algorithms. With enough cost budget (40k units) or with sufficient large or dense network (e.g. edge density of 12% or network nodes of 300), `GG` can achieve average generation latency as low as 4 milliseconds, while the `Non-SLs` algorithm incurs around 100-200 milliseconds. While the `Naive` and `CLUS` schemes perform similarly, the `CLUS` scheme achieves its performance by selecting more appropriate `SLS` (as is evident in Fig. 8); this is not surprising, as we limit the candidate super-links in the `Naive` scheme.

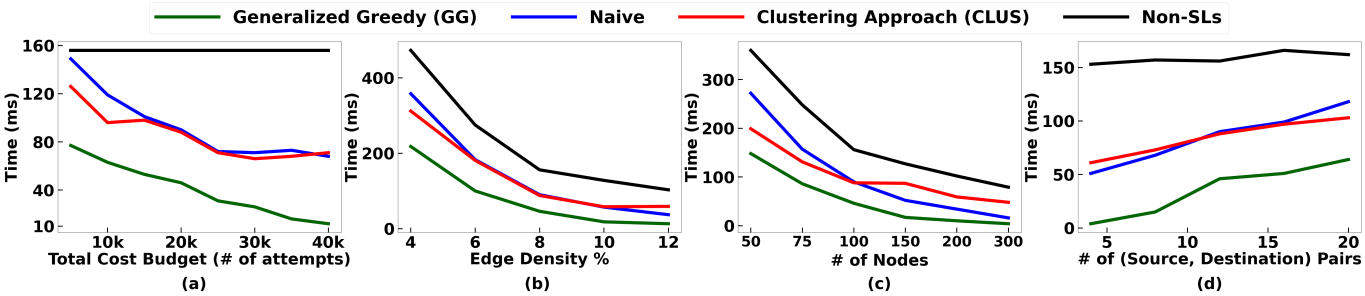

Fig. 6: Average generation latency over (s, d) EP requests by various algorithms for varying parameters.

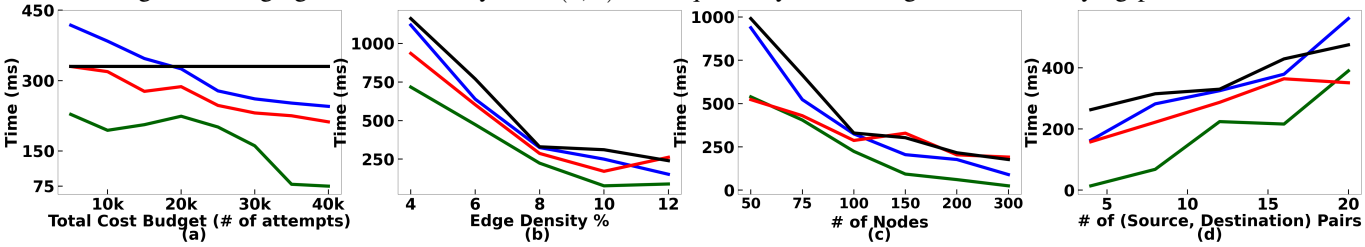

Fig. 7: Maximum generation latency over (s, d) EP requests by various algorithms for varying parameters.

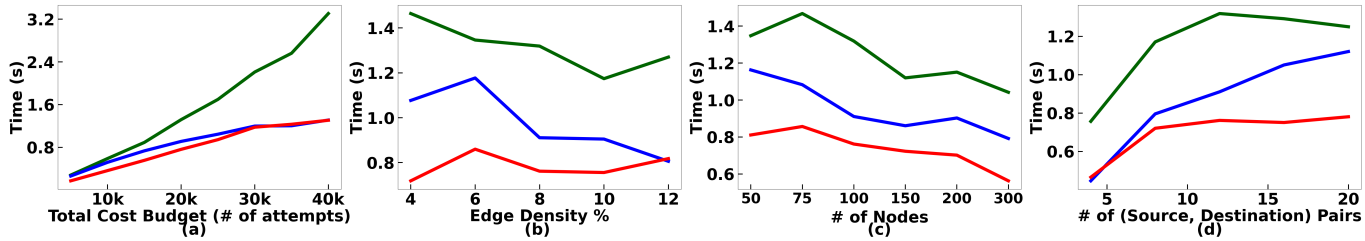

Fig. 8: Aggregate generation latency of super-link EPs incurred by various algorithms for varying parameters.

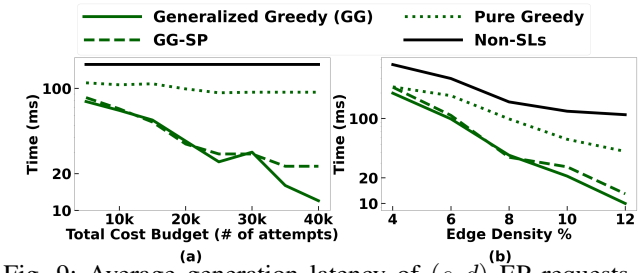

Fig. 9: Average generation latency of (s, d) EP requests, by certain greedy variants.

Fig. 7 plots the maximum (rather than average) EP generation latency over an (s, d) pair. Here, we observe that the algorithms are not able to sufficiently decrease the latency of all the EP requests; this is expected as the optimization objective of the algorithms is quite different. Nevertheless, we observe that our main algorithm `GG` still continues to outperform the other algorithms, and in particular, is able to lower the generation latency compared to the non-superlink scheme `Non-SLs`.

Lastly, in Fig. 8, we plot the aggregate latency of the EPs generated over the SLs selected by various algorithms; in some sense, this metric represents how efficiently each algorithm is able to utilize the cost budget. We observe that while `Naive` and `CLUS` results indicate under-utilization of the cost budgets, the `GG` algorithm is able to utilize the available network resources more effectively (taking into consideration, its high performance in the previous two plots). We stipulate that this high-utilization is due to the fact that we allow `GG` to also select non-shortest paths between node pairs.

Variants of Greedy Algorithm. We now consider two variants of our main Greedy algorithm `GG`. See Fig. 9. First, we observe that `GG-SP`, which is same as `GG` except that it only considers shortest-paths as associated paths for super-links, performs similarly than our `GG` algorithm when the cost budget is not high enough ($\leq 30K$), but the performance gaps suddenly increases with higher cost budget. Intuitively, shortest paths are preferred for super-links as they yield the same reduction in generation latency of (s, d) pairs with lower cost. But, with increased cost budget, as more and longer super-links get selected, the intersections between candidate super-links increase and selection of non-shortest paths become important. We also consider another variants of the greedy algorithm, viz., `Pure-Greedy`, which is a pure-greedy algorithm in the sense that it does not delete any already-picked super-link; thus, at each stage, it only considers super-links that are disjoint with the already picked super-links. We observe that `Pure-Greedy` significantly under-performs our main algorithm `GG`; this observation validates the strategy of using both *Update* and *Append* options within `GG`.

IX. Conclusions and Future Directions

In this paper, we proposed the approach of pre-distributing EPs to lower generation latency of expected EP requests, and addressed the key optimization problem in this context. Our future work is focused on investigated more sophisticated variants and settings of the `SLS` problem, e.g., combining super-links and multiple-paths approaches, allowing multiple super-link in each entanglement path.

References

[1] 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," *arXiv preprint arXiv:2010.12535*, 2020.

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

[3] J. Gambetta, "IBM's Roadmap For Scaling Quantum Technology," https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/, 2020.

[4] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, "Quantum internet: from communication to distributed computing!" https://arxiv.org/abs/1805.04360, 2018.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, 1982.

[12] D. Dieks, "Communication by EPR devices," *Physics Letters A*, vol. 92, no. 6, pp. 271–272, Nov. 1982.

[13] 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.

[14] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, "Long-distance quantum communication with atomic ensembles and linear optics," *Nature*, 2001.

[15] N. Sangouard, C. Simon, H. De Riedmatten, and N. Gisin, "Quantum repeaters based on atomic ensembles and linear optics," *Reviews of Modern Physics*, 2011.

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

[17] M. Uphoff, M. Brekenfeld, G. Rempe, and S. Ritter, "An integrated quantum repeater at telecom wavelength with single atoms in optical fiber cavities," *Applied Physics B*, vol. 122, no. 3, pp. 1–15, 2016.

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

[19] 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.

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

[21] K. Chakraborty, F. Rozpedek, A. Dahlberg, and S. Wehner, "Distributed routing in a quantum internet," 2019, https://arxiv.org/abs/1907.11630.

[22] S. Muralidharan, L. Li, J. Kim, N. Lütkenhaus, M. D. Lukin, and L. Jiang, "Optimal architectures for long distance quantum communication," *Scientific reports*, vol. 6, no. 1, pp. 1–10, 2016.

[23] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, jun 2013. [Online]. Available: https://doi.org/10.1088/0034-4885/76/7/076001

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

[25] W. Kozlowski, A. Dahlberg, and S. Wehner, "Designing a quantum network protocol," in *CoNEXT*, 2020.

[26] A. Meyerson and B. Tagiku, "Minimizing average shortest path distances via shortcut edge addition," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2009, pp. 272–285.

[27] Y. Dodis and S. Khanna, "Design networks with bounded pairwise distance," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, pp. 750–759.

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