Received 31 October 2023; revised 9 December 2023; accepted 18 December 2023. Date of publication 20 December 2023; date of current version 10 January 2024.

Digital Object Identifier 10.1109/OJCOMS.2023.3345166

REDiP: Ranked Entanglement Distribution Protocol for the Quantum Internet

LEONARDO BACCIOTTINI [1,2] (Graduate Student Member, IEEE), LUCIANO LENZINI [2], ENZO MINGOZZI [2] (Member, IEEE), AND GIUSEPPE ANASTASI [2]

[1] Dipartimento di Ingegneria dell'Informazione (DINFO), University of Florence, 50121 Florence, Italy
[2] Dipartimento di Ingegneria dell'Informazione, University of Pisa, 56122 Pisa, Italy

CORRESPONDING AUTHOR: L. BACCIOTTINI (e-mail: leonardo.bacciottini@phd.unipi.it)

This work was supported by the Italian Ministry of Education and Research (MIUR) in the framework of the ForeLab Project (Departments of Excellence).

ABSTRACT The distribution of entangled qubit pairs to end nodes is the key requirement of a quantum network to enable qubit state transmission through quantum teleportation. Existing protocols for entanglement distribution fix a specific swapping order on the involved quantum repeaters and delegate entanglement purification to upper-layer protocols. This limitation is problematic because entangled states tend to degrade due to quantum noise, and they cannot be purified if their fidelity (i.e., quality) falls below a certain threshold. It is therefore of the utmost importance to co-plan entanglement swapping and purification to achieve a target end-to-end fidelity. In this work, we present the Ranked Entanglement Distribution Protocol (REDiP), which overcomes the aforementioned limits by including the "ranks" mechanism to configure the ordering of both purification and entanglement swapping steps. We show how REDiP can easily be configured to implement custom entanglement swapping and purification strategies, including (but not restricted to) those adopted in two recent works. We also propose an algorithm to estimate the bandwidth to allocate on every link of the REDiP path, and we provide a set of guidelines on how REDiP ranks can be configured depending on user requirements and hardware configuration. Such guidelines are driven by original insights into purification performance. We conduct simulations to verify our results and assess the impact of different REDiP configurations on the performance of a repeater network, in terms of throughput and fidelity.

INDEX TERMS Entanglement distribution, entanglement purification, entanglement swapping, quantum Internet, quantum network protocol, quantum repeater networks.

I. INTRODUCTION

ENTANGLEMENT is a phenomenon typical of quantum mechanics where two or more systems (in our case, qubits) become so intertwined that their state must be described as a single, global entity, even if the systems are spatially separated. Quantum networks exploit shared entangled states and classical communication infrastructures to enable a whole new set of disruptive applications, such as distributed quantum computing [1], blind quantum computing [2], quantum key distribution [3], [4], and quantum secret sharing [5].

The key challenge at this early stage of quantum networks is to generate entangled qubit pairs locally and distribute them to the end nodes of the network. The problem is made much harder by the no-cloning theorem of quantum mechanics, which prevents us from employing classical repeaters to propagate quantum information stored into photons. A solution to this problem is the adoption of quantum repeaters to distribute entanglement over physical links and then propagate it through a protocol called *entanglement swapping*. Quantum noise plays another crucial role as it introduces errors in both entanglement generation and swapping protocols. Failure to adequately mitigate quantum noise results in the exponential degradation of entanglement fidelity—a metric indicating the quality of entanglement—over both time (quantum memory decoherence) and distance (subsequent entanglement swaps). Early-stage quantum networks, which will likely employ

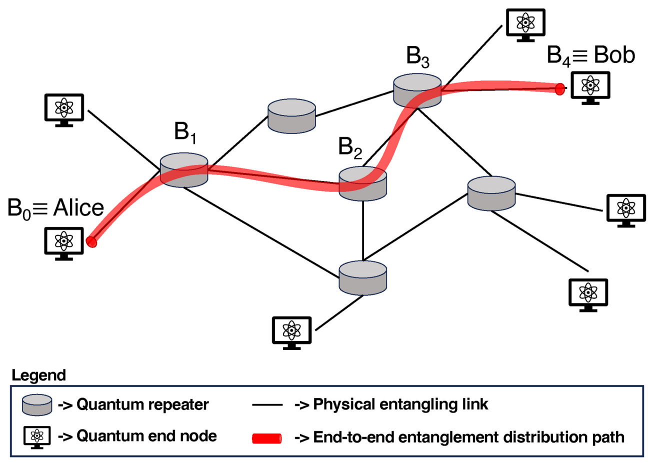
B₃
B₄≡ Bob
B₁
B₂
B₀≡ Alice
Legend
-> Quantum repeater
-> Physical entangling link
-> Quantum end node
-> End-to-end entanglement distribution path

FIGURE 1. The system model. A path $B = [B_0, B_1, B_2, B_3, B_4]$ is determined to connect two end nodes Alice (B_0) and Bob (B_4) of the quantum network. The entangled qubit pairs generated on the physical links of the path are employed to distribute entanglement shared among Alice and Bob.

first-generation quantum repeaters [6], can restore the fidelity of entangled qubits through a probabilistic procedure called *entanglement purification* (or distillation). In the future, as the technology evolves, it will also be possible to employ quantum error correction techniques [7], but these solutions are not achievable with near-term hardware [8].

Several architectures for a Quantum Internet have been proposed over the last few years [9], [10], [11], [12], and they all share the common goal of achieving entanglement distribution to end nodes through combinations of entanglement swapping and purification, following a scheme called *Purify-and-Swap* [13].

We show the model of our system in Fig. 1. A path is established on the quantum network to connect two (quantum) end nodes, Alice and Bob, willing to share entanglement. Entangled qubit pairs are continuously generated as a background process on the physical links along the path, which may exploit diverse media such as optics fiber or open space. The goal of this work is to describe the *Ranked Entanglement Distribution Protocol* (REDiP): a configurable protocol that schedules and carries out the *Purify-and-Swap* operations required to propagate entanglement to the interested end nodes.

Entanglement swapping is a two-step protocol involving a Bell-state measurement (BSM) followed by the transmission of a final measurement outcome, namely two classical bits, so that the recipient is able to determine in which of the four maximally-entangled states the qubit pair has landed. This requirement introduces a new problem, that is determining the order in which this classical synchronization between repeaters takes place. Such a decision is very important because it defines –and is defined by- the order of performing entanglement swapping among nodes. With reference to Fig. 1, the three repeaters B_1, B_2, B_3 may carry out their BSM in parallel, consecutively, or in any other order. We call this choice *Entanglement Swapping Strategy* (ESS).

Entanglement purification can be carried out at any time on two or more entangled qubit pairs shared between the same nodes. Its outcome is a smaller number of pairs with higher fidelity than the input ones, with a certain success probability. If we include the possibility of performing purification in our discussion, we define the *Purification and Entanglement Swapping Strategy* (PESS) as the policy that determines both the order of entanglement swapping other than when and where (i.e., at what node) purification rounds are to be carried out.

In this work, we extend the presentation of REDiP from [14]. The main contributions of this work with respect to [14] can be summed up as follows: (i) providing additional details about REDiP design, (ii) implementing an algorithm to estimate the bandwidth (*pairs/s*) that a REDiP instance requires on every link of the path connecting the end nodes, and (iii) describing a set of hands-on guidelines to configure REDiP ranks so that a target end-to-end fidelity is achieved with the least amount of resources. This last contribution is supported by our insights on the performances of the popular DEJMPS (from the authors' last names) purification protocol [15]. Our simulation campaign evaluates the performances of different PESSs implemented on REDiP, showcasing the impact of different sources of quantum noise on the resulting throughput and fidelity. Results show that REDiP configurability is essential to meeting user requirements in a quantum network subject to quantum errors and link conditions that may change over time and space.

This work is structured as follows: Section II gives an overview of the main concepts behind REDiP and how it can be configured, then Section III describes the protocol with a more technical and detailed approach. Section IV shows the algorithm used to estimate the bandwidth on every link of the path. Section V provides the guidelines on how REDiP connections can be configured to meet specific user requirements. Section VI shows the results of a simulation campaign that evaluates the performances of different PESSs implemented on REDiP under varying scenarios. Finally, Section VII draws some final remarks and future work.

A. RELATED WORK

A major distinction arises in the literature between two architectural model classes for quantum networks, which we divide into *time-slotted* and *asynchronous*. Time-slotted architectures, from which we point out [12], [16], [17], [18], assume that the whole quantum network evolves according to discrete time slots of a fixed duration. A centralized or distributed clock-sharing procedure is therefore employed by all these proposals. Such a time-slotted approach introduces a significant simplification in the management of entanglement distribution: At each time slot, the control plane (either centralized or distributed) evaluates which quantum network nodes currently share entangled qubit pairs, and a *virtual graph* is generated where every edge represents an available entangled qubit pair. Within the same time slot, the control plane schedules the requests to serve and routes every request for end-to-end entanglement over a dedicated path on the virtual graph. The algorithms used for scheduling and routing requests depend on the solution; we point out the work by Cicconetti et al. [16], which features a very clear formulation of the problem.

The advantage of time-slotted architectures is that they can be very responsive: requests for a reasonable amount of end-to-end entangled qubits can often be served in a single time slot. Contrarily, their intrinsic drawback is the under-utilization of network resources: Entanglement generation is attempted only at the beginning of every time slot, leading to a loss in the overall throughput in exchange for easier management of available resources. Some works such as [18] try to partially counter this downside by accumulating entangled qubit pairs on specialized quantum network devices, and making them available when needed. An additional critical challenge is that the maximum duration of the time slot is given by the quantum memories coherence time, since entangled qubits are allocated at the beginning of a time slot and delivered to the application at the end of the same time slot (in the best case).

On the other hand, asynchronous architectures like [10], [11], [19] adopt a greedier approach where entanglement generation on every physical link is considered a continuous, stochastic process that delivers new entangled qubit pairs at an average throughput over time. Protocols designed for such architectures do not work within a specified time slot and must face the possibility that not enough entangled qubits may be available at a given time. The consequence is that quantum network protocols for entanglement distribution must be robust to the temporary unavailability of qubits, and feature a higher complexity in terms of control and synchronization messages across different nodes. Asynchronous quantum networks are thus potentially able to guarantee a higher utilization of the network resources, but the overhead likely required to reconfigure the quantum network and serve new scheduled requests can be very high.

Overall, we expect time-slotted quantum networks to be effective for short-lived applications that require a small number of end-to-end entangled qubits, thus privileging *responsiveness*. Asynchronous quantum networks promise instead to be optimal for applications that require long streams of entangled qubits over time (e.g., to share several secrets using quantum key distribution), thus privileging *throughput*.

Some connection-oriented protocols for entanglement distribution, suitable for asynchronous architectures, have been recently proposed in the literature, from which we point out [20], [21], [22]. One of the limits of these previous works is that they are tightly bonded to a single ESS, i.e., a specific order in which entanglement swapping is carried out among the nodes along the path.

The three different ESSs adopted in these works are shown in Fig. 2 on a five-node path: (i) *Consecutive ESS* (Fig. 2(a)), employed in [20], where a node performs its BSM only after receiving the outcome from the previous node on the path and then sends the outcome to the next node. (ii) *Nested ESS* (Fig. 2(b)), employed in [21], where the BSM ordering is recursively determined by extracting the nodes of the path in odd positions -with positions starting from zero- until only the end nodes remain. This definition means that B_1 and B_3 can perform the BSM concurrently, and B_2 must wait to receive the outcomes from both B_1 and B_3. Note that this strategy is applicable only if the number of nodes can be written as $2^n + 1$ for some $n \in \mathbb{N}$. (iii) *Parallel ESS* (Fig. 2(c)), employed in [22], where all repeaters perform their BSM concurrently and transmit the outcome directly to at least one of the end nodes.

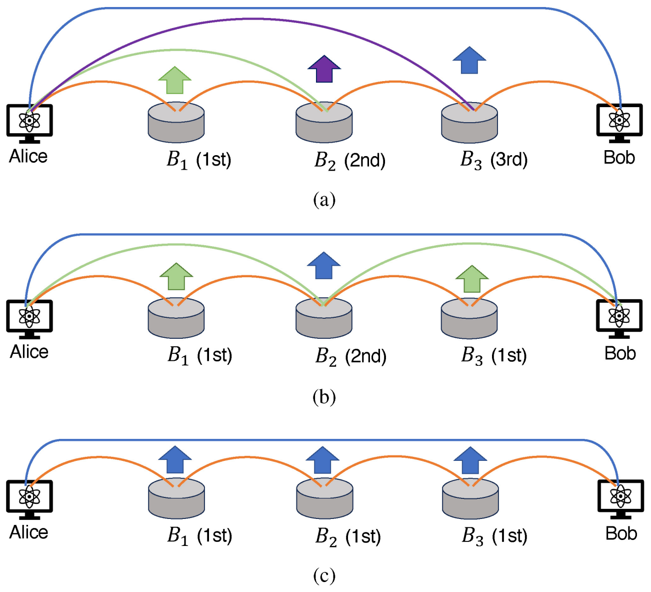

FIGURE 2. Three entanglement swapping strategies on a five-node path (two end nodes and three repeaters): (a) Consecutive, (b) Nested, (c) Parallel. The arcs represent entanglement between the two connected nodes, whereas the arrows and the legend specify the entanglement swapping order.

More specifically, Kozlowski et al. protocol [22] exploits Parallel ESS to counter the effect of low memory coherence times. The proposal by Li et al. [20] features a detailed description of the connection establishment phase, involving a resource allocation mechanism based on quantum memory slots partitioning. Their protocol uses Consecutive ESS to have strict control over the timing of the actions performed by each node. Finally, the proposal by Aparicio et al. [21] is not exactly a protocol, but a recursive protocol stack that schedules entanglement swapping and purification as separate protocols, one on top of the other.

Another key limitation of these existing protocols is that they do not support entanglement purification as an integrated mechanism. Together with the forced ESS, this leads to a lack of flexibility when it comes to meeting user requirements in terms of fidelity and throughput. Of course, it is always possible to have a recursive protocol stack where several connections and purification protocols are installed one on top of the other to implement an arbitrary

PESS, as done in [21], [23] and suggested in [22], but this inevitably introduces additional overhead and complicates network reconfiguration, which is already one of the critical problems of asynchronous solutions.

REDiP is a novel protocol for entanglement distribution in asynchronous quantum networks. It incorporates these state-of-the-art protocols, as all of them can be implemented as special REDiP configurations. Moreover, REDiP enables a whole set of custom strategies that would not be achievable with existing protocols. This result is obtained through the "ranks" mechanism, which enables an arbitrary ESS and includes entanglement purification as a native feature, which was not taken into account within the most recent protocols. Moreover, the same REDiP connection can potentially be dynamically reconfigured to adapt to changes in requirements and failures, without having to dismantle and set up a new connection.

II. PROTOCOL OVERVIEW

REDiP is a connection-oriented protocol. Therefore, a connection on a path between the two end nodes must be created before distributing the entanglement. This connection conveys classical bits and is provided by a conventional infrastructure (for example, the classical Internet). To avoid ambiguities with entangled connections, we will refer to this preliminary connection as *tunnel*.

Taking as reference the quantum protocol stack introduced in [10], REDiP is placed at the *Network layer*. The underlying *Link layer* has the task of generating heralded entangled Bell pairs on physical links that are consumed by Network layer protocols to distribute end-to-end entanglement. We will refer to upper layer protocols as *Users* of REDiP.

Finally, we assume that the quantum network architecture where REDiP is used has an addressing scheme where each entangled qubit pair has an identifier shared at least among the nodes holding the two ends of the pair. For the sake of clarity, we call an entangled qubit pair between adjacent nodes an *entangled link* or just a *link*. We also use the term *entangled segment* or just *segment* when the entangled pair resides on non-adjacent nodes. Finally, we call *end-to-end entangled connection* or just *connection* an entangled segment that resides on the end nodes. Clearly, the two qubits of an entangled pair embody the *endpoints* of a link, segment, or connection.

REDiP comprises three consecutive phases, namely (i) *establishment*, (ii) *active*, and (iii) *closing*. The next sections will give an overview of all the tasks carried out during each phase.

A. ESTABLISHMENT PHASE

The establishment phase is the preliminary sequence of actions taken by the nodes where they agree to be part of a REDiP tunnel. During this phase, nodes also determine the bandwidth (average *pairs/s*) to be reserved for the tunnel on each physical link of the path. As shown in Fig. 3, The establishment phase starts at an end node A when a user, typically an upper layer protocol, submits a NEW_TUN message that is forwarded to all nodes on the path toward the other end node B. The NEW_TUN message contains the identifiers of all the nodes on the path, together with a set of parameters used to configure the connection (e.g., ranks). We also include an optional confirmation phase where a TUN_ACK message is propagated back to the originating end node A to confirm the tunnel establishment. When the TUN_ACK reaches the A node, all nodes reserve the agreed bandwidth on their physical links, and entanglement generation can start. This approach is borrowed from classical Internet protocols like the Resource Reservation Protocol (RSVP). The format of the aforementioned messages is provided in the technical part, Section III.

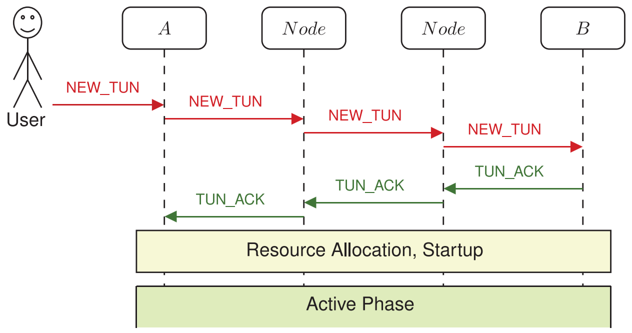

FIGURE 3. Simplified example of REDiP tunnel establishment.

B. ACTIVE PHASE

During the active phase, the REDiP tunnel continuously generates end-to-end connections and delivers them to the users. Before delving into the operational description, we introduce the concept of *ranks* and how they can be used to configure the tunnel and suit user requirements. Each node belonging to the REDiP tunnel is assigned a rank, namely an integer value. Ranks can assume values between 0 and $N-1$, where N is the total number of nodes. Ranks are used to determine the order in which entanglement swapping is carried out: nodes with a lower rank swap earlier, whereas two nodes sharing the same rank can swap concurrently. By design, we impose that the end nodes of the tunnel share the highest rank. The ranks of all the nodes are grouped inside a rank vector R, which is distributed during the establishment phase. As an example, we implement the three ESSs from Fig. 2. REDiP realizes the three strategies by assigning the rank vectors $R = [3, 0, 1, 2, 3]$, $R = [2, 0, 1, 0, 2]$, and $R = [1, 0, 0, 0, 1]$ respectively.

The user can also add *entanglement purification*. We assume DEJMPS protocol [15] (from the authors' last names) to be used as the default purification protocol, as it has been proven to be optimal for Werner states (worst case scenario). The user can add an arbitrary number of purification rounds to be performed at each rank. In the example of Fig. 4, we can see the purification vector $P = [1, 1, 0]$ applied on the Nested ESS. Such assignment configures REDiP so that 0-ranked and 1-ranked nodes apply one purification round

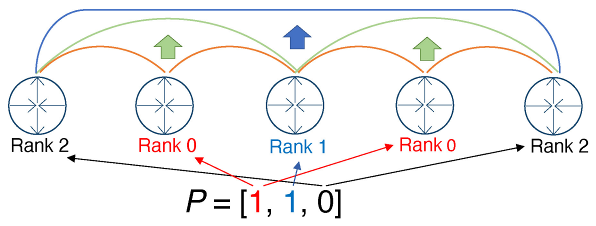

FIGURE 4. Example of purification assignment on the Nested ESS.

to all qubit pairs before they can be swapped. Contrarily, 2-ranked nodes (i.e., the end nodes) do not perform any purification on the end-to-end connections.

The active phase terminates when a certain number K of end-to-end connections, pre-determined during the establishment phase, has been delivered and consumed by the user. During the *closing phase*, nodes free up the allocated resources, and the tunnel is dismantled.

III. REDIP TECHNICAL DESCRIPTION

In this section, we delve into the design of REDiP, comprising message formats, internal states, and flows of communication. The main focus of this section is the active phase, which is the core of REDiP operation.

We define some reference variables, called *REDiP parameters*, that will be used across this section:

- N is the number of nodes composing the *REDiP path*.
- B is the vector of node identifiers along the REDiP path, in the order they are traversed. B_i indicates the *ith* node on the path, $i \in \{0, \dots, N-1\}$.
- R is the *REDiP rank vector*, where each element R_i is an integer indicating the rank of node B_i.
- P is the *REDiP purification vector*, where each element P_r is an integer indicating how many purification rounds must be performed by nodes whose rank is r, $r \in \{0, \dots, max\{R_i \in R\}\}$.
- K is the number of end-to-end pairs that a given tunnel must deliver to the destination end nodes.

Furthermore, as in [22], we assume that the Link layer protocol continuously generates entangled links once the tunnel is open. For any given node, from now on we will refer to the directions towards the rightmost and leftmost nodes on the path L as *upstream* and *downstream* respectively.

For what concerns the REDiP establishment phase, its functionality has been fully unraveled in the previous section. We are now ready to briefly complete its description by defining the information conveyed within NEW_TUN and TUN_ACK messages. The NEW_TUN message is initialized by the user and contains the REDiP parameters (B, R, P, K) and a unique identifier for the REDiP tunnel. Additionally, every node that receives and forwards the NEW_TUN message appends the fidelity and the maximum bandwidth available on the attached physical links (expressed in generated *pairs/s*), and its hardware noise parameters, which can be represented by the accuracies (i.e., the success probabilities) of unary, binary quantum gates, and measurement apparatus. When the last node B_{N-1} receives the NEW_TUN message, it has all the information required to derive the bandwidth to be allocated on all links of the path. The algorithm used to derive the bandwidth is discussed in Section IV.

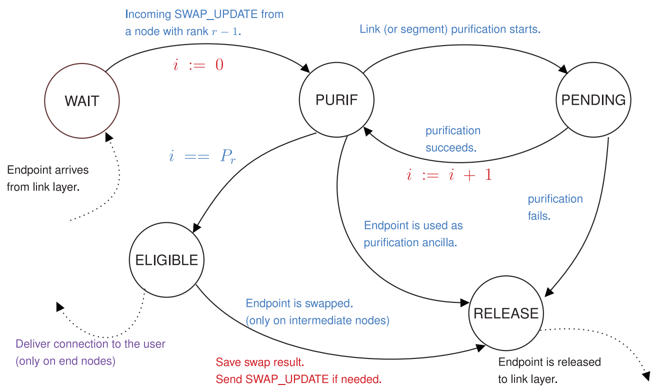
Incoming SWAP_UPDATE from a node with rank $r-1$.
$i := 0$
Link (or segment) purification starts.
WAIT
PURIF
PENDING
purification succeeds.
$i := i + 1$
$i == P_r$
Endpoint arrives from link layer.
purification fails.
Endpoint is used as purification ancilla.
ELIGIBLE
RELEASE
Endpoint is swapped. (only on intermediate nodes)
Deliver connection to the user (only on end nodes)
Save swap result. Send SWAP_UPDATE if needed.
Endpoint is released to link layer.

FIGURE 5. Finite state machine for an endpoint on a node with rank *r*. Blue labels indicate the conditions for the state transition, whereas red labels indicate the action performed during the transition.

The TUN_ACK message is initialized by the last node B_{N-1} with the REDiP tunnel identifier, the bandwidth to be allocated on each link of the path, and a timestamp indicating the time at which the active phase may start. This timer does not assume a tight synchronization between nodes, it is used to prevent nodes from entering the REDiP active phase before the TUN_ACK message has been forwarded back to the first node B_0.

The control operations of the REDiP active phase are carried out by three classical messages called (i) SWAP_UPDATE, used to notify nodes about the result of an entanglement swapping, (ii) PURIF_SOLICIT and (iii) PURIF_RESPONSE used during the entanglement purification procedure. We will gradually go through their definition and usage as the description moves forward.

A. ENDPOINT LOGICAL STATES

During the active phase, Link layer protocols continuously generate and deliver new endpoints (i.e., entangled qubits) to REDiP on every node along the path. At a given time, as shown in the state diagram of Fig. 5, every endpoint held by an r-ranked node of the REDiP path can be in one of the following possible *logical states*:

- WAIT is the initial state for endpoints signaled by the Link layer when an entangled qubit pair is generated.
- PURIF is immediately entered if $r = 0$, otherwise it is entered when a SWAP_UPDATE message for this endpoint arrives from a node whose rank is $r - 1$. An endpoint stays in this state if the link, segment, or connection must be purified.
- PENDING is entered when an endpoint is submitted to the purification procedure. If the procedure succeeds, the state transitions back to PURIF and the purification counter i is increased by one.

- RELEASE is the transitional state of endpoints that are about to be released back to the Link layer. It is reached when the endpoint is used as an ancilla for purification, when the endpoint is swapped, or when purification fails.
- ELIGIBLE is the state for endpoints that are ready for entanglement swapping (on intermediate nodes), or that are ready to be delivered to the user (on end nodes). It is entered when the endpoint in PURIF state has been successfully purified a number of times equal to P_r.

B. SWAP UPDATES

As soon as the REDiP tunnel has been established, each intermediate node B_i processes the parameters (B, R, P, K) and infers the following variables:

- B_i Swapping Destinations (SD^i), a two components vector where its components SD_0^i and SD_1^i are picked as the downstream and upstream nodes closest to B_i whose rank is *strictly* higher than R_i,
- B_i Swapping Neighbors (SN^i), a two components vector where its components SN_0^i and SN_1^i are picked as the downstream and upstream nodes closest to B_i, whose rank is *equal or* higher than R_i.

SWAP_UPDATE messages share some similarities with the TRACK message from [22], in the sense that they serve the purpose of keeping track of what endpoints have been swapped and the final Bell state of the new segment. The difference lies in the fact that TRACK messages are always forwarded by all nodes on the path towards an end node, whereas SWAP_UPDATEs are generated and forwarded by swapping neighbors, and delivered to a swapping destination which may or may not be an end node. The idea is that REDiP ranks partition the path into several nested chains of swapping neighbors. At any time, there exist multiple SWAP_UPDATEs running at different ranks, which are forwarded upstream or downstream by swapping neighbors toward a swapping destination.

An intermediate node B_i generates a SWAP_UPDATE message after the execution of the entanglement swapping only if $SN_0^i \equiv SD_0^i$ or $SN_1^i \equiv SD_1^i$, i.e., only if at least one of its swapping neighbors is also a swapping destination.

Fig. 6(a) shows the behavior of an r-ranked node B_i when its swapping neighbors and destinations do not coincide. We can see that B_i receives an upstream SWAP_UPDATE from SN_0^i (time T_0), but it has not swapped its two local endpoints yet, so the message cannot be immediately forwarded. As soon as B_i has an upstream endpoint in ELIGIBLE state, it is swapped with the endpoint specified by the received message (time T_1). The final Bell state measurement is collected and saved in a temporary record as it is done in [22], and the two swapped endpoints transition to the RELEASE state. At this point B_i can add the stored measurement result to the upstream SWAP_UPDATE, which is forwarded to SN_1^i (time T_2). When B_i receives the SWAP_UPDATE from SN_1^i, it immediately updates and forwards downstream the message (time T_3). When both SWAP_UPDATEs are delivered to the corresponding swapping destination, the two endpoints of the new segment between SD_0^i and SD_1^i transition to the state PURIF (time T_4). Fig. 6(b) shows the complementary case where the upstream swapping neighbor and destination coincide. The key difference with the previous example is that after the swap (time T_1), B_i must generate the downstream SWAP_UPDATE and send it to SN_0^i (time T_3). Of course, in the mirror case of downstream coincidence ($SN_0^i \equiv SD_0^i$), then B_i would generate and send the upstream SWAP_UPDATE instead of the downstream one.

Now that we sketched the role and the information conveyed by SWAP_UPDATEs, we describe their format. When it is generated, a SWAP_UPDATE is meant to be delivered to a specific swapping destination node, which is marked within the message as its *target*. The other swapping destination is instead marked within the message as its *origin*. The SWAP_UPDATE is forwarded to one of the swapping neighbors, marked within the *next-hop* field. The BSM measurement outcome is encoded by means of two message fields, namely (i) the identifier of the swapped segment (*segment_id* field), which is shared with the next-hop, and (ii) the new quantum state of the EPR pair (*new_state* field), to discriminate in which of the four Bell states the segment has landed after the measurement. These last three fields are updated every time that the SWAP_UPDATE is received and forwarded so that when it reaches its target node, it is able to determine which endpoint should transition to the PURIF logical state, as well as the final Bell state of the entangled segment.

C. PURIFICATION

What is still missing to generate connections is the link (or segment) purification procedure that takes endpoints from PURIF state and eventually turns them into ELIGIBLE. Even though REDiP is not intrinsically bound to a specific purification protocol, from now on we consider the DEJMPS [15] (from the authors' last names) as the default purification protocol employed by REDiP, which has been proven optimal when the purified pairs are isotropic Werner states (worst case scenario). Since we cannot make any assumption on the quantum hardware installed by REDiP nodes, DEJMPS is the standard choice. The inexpert reader may find details about DEJMPS in Section IV.

In REDiP each rank has a dedicated *purification step*. Every purification step can be composed by zero, one or more *purification rounds*, as specified by the REDiP purification vector P. The number of purification rounds determines how many consecutive times the endpoints of an r-ranked node have to (successfully) go through the purification protocol before they can reach the ELIGIBLE state. In the following, we show the purification procedure employed by REDiP nodes to realize a purification step.

The purification procedure can be carried out between two swapping neighbors or between a node and one of its swapping destinations. In the latter case, the endpoint

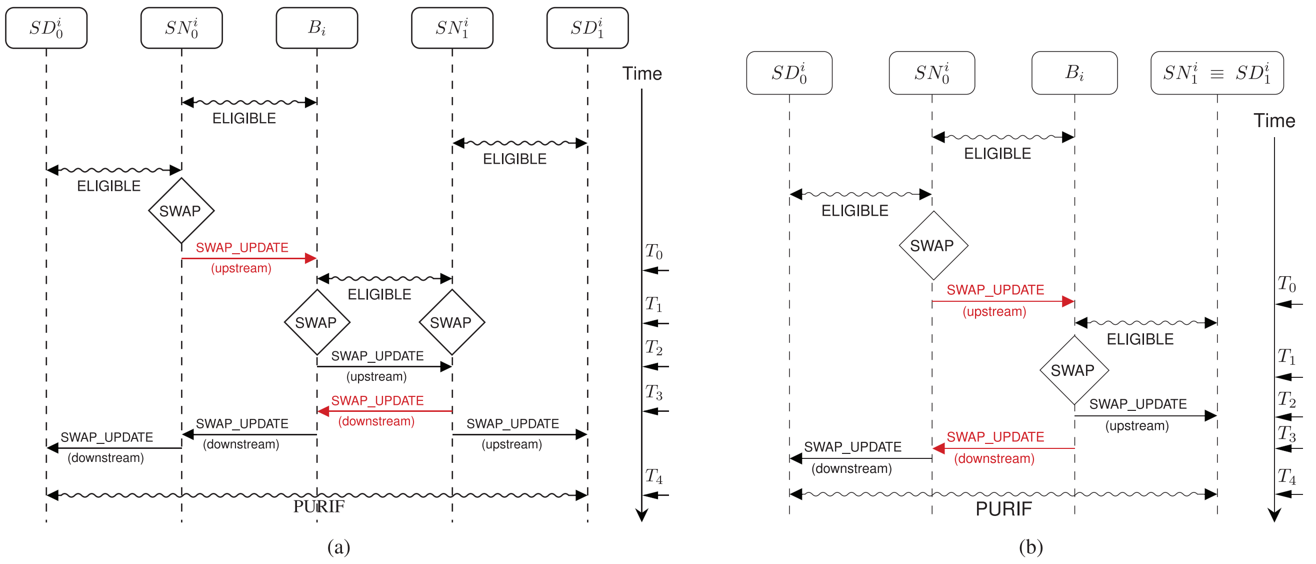

FIGURE 6. Example sequence of REDiP when (a) swapping neighbors and destinations are distinct and (b) upstream swapping neighbor and destination coincide. Waved arrows indicate that endpoints of an entangled link (or segment) between the two nodes have transitioned to a new state. The generation of a SWAP_UPDATE message is highlighted in red.

residing on the swapping destination starts in WAIT state instead of PURIF and it never transitions to ELIGIBLE in case of successful purification. The two actors are called *Initiator node* and *Solicited node*, where the former has the initial task of choosing the link to purify and one other link as ancilla. The policy used for this choice depends on the adopted purification scheme. For example, a *recurrence* scheme requires the ancilla to have the same fidelity as the purified link, whereas an entanglement *pumping* scheme like the one proposed in [13] (which still uses the DEJMPS protocol under the hood) allows the exploitation of ancillas with a lower fidelity than the purified link. The Initiator node then executes the purification protocol and sends a PURIF_SOLICIT message containing the identifier of both the link to purify and the ancilla, and their purification measurement outcome. The Solicited node receives the message and applies the purification algorithm itself. If the measurement outcome matches the one within the PURIF_SOLICIT message, then the purification was successful. In any case, the Solicited node replies with a PURIF_RESPONSE message, containing the purified and ancilla link identifiers (piggyback approach), and the purification outcome (“ok” or “fail”). If the purification is successful, the purification counter i is increased by one, signifying that the link is ready for the next purification round, otherwise both endpoints are released back to the Link layer. Ancillas are always released to the Link layer right after their measurement. We show an example in Fig. 7, where pair A is purified and pair B is used as ancilla. The procedure is repeated until the purification counter i reaches the value P_r, where r is the rank of the Initiator node. To determine the actor roles there is a simple rule: the lowest ranked node is the Initiator. The tie-breaking rule is left up to REDiP implementation; one could simply set the

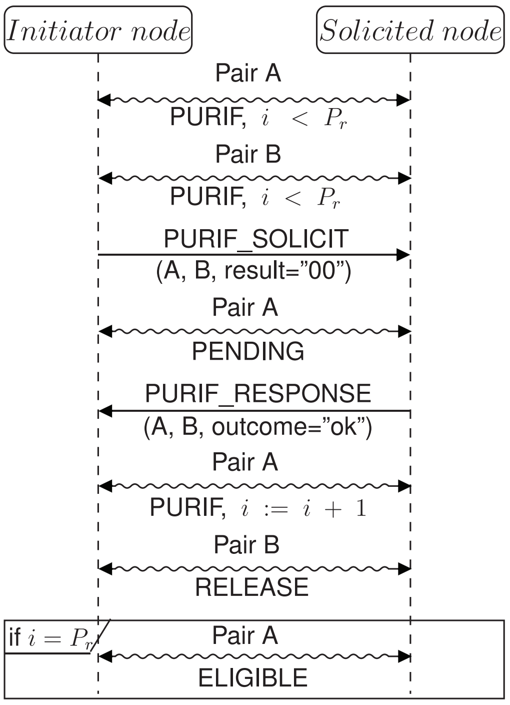

FIGURE 7. Example sequence of REDiP purification. Waved arrows indicate that both endpoints of the link (or segment) have transitioned to a new state. For the PURIF state, there is also the new value of the purification counter *i*.

downstream node to always be the Initiator when the ranks are equal.

D. IMPLEMENTING DIFFERENT STRATEGIES

In this section, we show how REDiP configurability allows to implement the Parallel ESS, Consecutive ESS, and Nested ESS. For each one of these strategies, we provide a way to compute the rank vector R given the number of nodes N.

1) CONSECUTIVE STRATEGY

The rank vector $R = [N-2, 0, 1, 2, \ldots, N-3, N-2]$ reproduces the Consecutive ESS from [20], where entanglement swapping is sequentially carried out by intermediate nodes along the path in the upstream direction. According to this REDiP version, it is not up to the downstream end node to trigger entanglement swapping on the successive repeater, which is instead the approach used in [20]. This saves one transmission hop for each SWAP_UPDATE, leading to a slight performance improvement.

2) NESTED STRATEGY

The Nested ESS is implemented by a rank vector R where each element R_i is defined as follows:

$$R_i = \max_{r \in \{0,1,\ldots,k\}} \{r \mid i \mod 2^r = 0\}, \tag{1}$$

where $N = 2^k + 1$ for some $k > 0$. This last condition is an assumption for the applicability of this strategy.

3) PARALLEL STRATEGY

If we set the ranks as $R = [1, 0, 0, \ldots, 0, 0, 1]$, we reproduce the Parallel ESS used in [22]. However, this REDiP configuration performs slightly better because SWAP_UPDATEs are generated by the nodes adjacent to the end node. TRACK messages from [22] are instead generated by the end nodes themselves. This saves one transmission hop for each SWAP_UPDATE, which translates into less idle time spent by qubits inside of the end nodes' quantum memories.

IV. ESTIMATING RESOURCES AND BANDWIDTH

REDiP nodes utilize NEW_TUN and TUN_ACK messages during the establishment phase to estimate the bandwidth (average *pairs/s*) that should be allocated on each link along which the connection is routed, so as to meet the user requirements in terms of target fidelity. In this section, we put forward the necessary information leveraged by nodes, along with an algorithm they can employ to find out the aforementioned bandwidth. The algorithm we propose is based on two initial assumptions: (i) The average entanglement generation rate on physical links does not depend on the quantum memory occupancy. (ii) Quantum memory decoherence is not taken into account. We will remove these two hypotheses in the simulation analysis, where we will delve into practical applications of this algorithm in a real-world scenario.

To start, let's establish the notion of *insistence*: A purification step, which encompasses one or more purification rounds, between two REDiP nodes denoted as A and B, is said to **insist** on a link denoted as l if the REDiP sub-path connecting A to B includes l.

The objective is to evaluate, for each link l within the REDiP path, the average consumption of entangled pairs resulting from all purification steps that insist on l. From now on, we will use the expression *resources consumed by a physical link*, which we define as the number of noisy entangled pairs required during all purification steps insisting on that physical link, to propagate a single end-to-end entangled pair with a target fidelity.

A. DEJMPS AND SWAPPING PRELIMINARIES

Every DEJMPS purification round is characterized by the initial fidelity F of the pairs that are subject to purification, a probability of success P_s, and a final fidelity F'. If F does not fall below a certain threshold, then $F' > F$. We implicitly assume that the pairs subject to purification are in a Werner state $\hat{\rho}$ with fidelity F

$$\hat{\rho} = \left(\frac{4F-1}{3}\right)|\phi_+\rangle\langle\phi_+| + \left(\frac{4-4F}{3}\right)\frac{\hat{I}}{4}, \tag{2}$$

where $|\phi_+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is the reference Bell state, and $\frac{\hat{I}}{4}$ is the completely mixed state. In general, the output state ρ' with fidelity F' remains diagonal in Bell basis after DEJMPS, but not as an isotropic Werner state [13].

The final fidelity F' and the success probability P_s can be computed through the formulas provided by Dür et al. [13], which are extremely verbose but not hard to evaluate. Both F' and P_s depend on F and the hardware noise parameters of the devices, namely binary quantum gates and measurement accuracies, referred to as p_2 and η respectively.

Given a purification step i, composed of K rounds, we can compute the average number of qubit pairs consumed to get a single distilled pair as [13]

$$M_i = \prod_{j=1}^{K} \frac{2}{P_s^j}, \tag{3}$$

where P_s^j is the success probability of the j-th round.

Regarding entanglement swapping, we can again apply equations from [13] to determine the output state after S consecutive swaps. If the quantum states of all swapped pairs are diagonal in Bell basis, the output state $\hat{\rho}_S$ is diagonal with a fidelity F_S lower than the fidelities of all swapped pairs. F_S can be computed as a function of all input states and the noise parameters of the involved devices. We report here the formula in the case all the S swapped pairs are in Werner states sharing the same fidelity F:

$$\left(\frac{4F_S-1}{3}\right) = \left(p_2\frac{4\eta^2-1}{3}\right)^{S-1}\left(\frac{4F-1}{3}\right)^{S}. \tag{4}$$

B. COMPUTE RESOURCES ON EACH LINK

The number of purification steps that insist on a link l of the REDiP path is at most equal to the highest REDiP rank. The algorithm we propose aims to compute the average number of resources (i.e., entangled qubit pairs) required on each link l in the REDiP path to establish a single end-to-end connection. The procedure is outlined in Algorithm 1. Leveraging the REDiP rank mechanism, it employs a bottom-up recursive strategy, initially focusing on purification and subsequently swapping on each rank.

The total resources consumed by purification steps insisting on every physical link of the path are computed and stored in a vector variable M, which is the output of the algorithm. The segments fidelity is also updated after every purification step. Following entanglement swapping, the algorithm is recursively called on a reduced set of nodes B', excluding nodes that have already participated in swapping. The exit condition is simple: if only the two end nodes remain, the end-to-end connection is ready.

The algorithm integrates three fundamental subroutines, employing Dür's formulas [13] as referenced in the previous section:

- *ResourcesDEJMPS:* computes the average resources needed for a successful purification step, given the initial fidelity, the number of purification rounds, and the noise parameters. It applies (3). Complexity: $\mathcal{O}(1)$
- *FidelityDEJMPS:* computes the final fidelity of a purification step, has the same arguments as *ResourcesDEJMPS*. Complexity: $\mathcal{O}(1)$
- *FidelitySwap:* computes the final fidelity after entanglement swapping, given the list of swapping nodes, the initial fidelity of every segment, and the noise parameters. Complexity: $\mathcal{O}(\textit{number of swapping nodes})$

The overall complexity of Algorithm 1 is given by the first loop, as it potentially iterates over all the links of the REDiP path. In the worst-case scenario (corresponding to the Consecutive ESS), the algorithm involves $\mathcal{O}(N)$ recursive calls, so we can state that its complexity is at most $\mathcal{O}(N^2)$. Anyway, since the number of hops of a network path is expected to be at most in the order of a hundred nodes, we can safely assume the algorithm to be computationally easy to solve.

C. DERIVE THE BANDWIDTH

Now that we are able to compute the amount of resources that are needed to generate a single end-to-end connection, we apply a trivial water filling optimization to derive the bandwidth to allocate on each physical link. The optimization problem is as follows:

$$\textbf{maximize } t \tag{5}$$

$$\textbf{subject to:}$$

$$r_l = t * M_l \;\forall l \in L \tag{6}$$

$$r_l \leq r_l^{max} \;\forall l \in L \tag{7}$$

where L is the set of all physical links on the REDiP path, the quantities M_l are the elements of the M vector computed with Alg. 1, and r_l^{max} is the maximum bandwidth that we can allocate on that link, which is given by hardware parameters. The solution of this problem can be obtained analytically as

$$t_{opt} = \min_L \left\{ \frac{r_l^{max}}{M_l} \right\}, \tag{8}$$

where t_{opt} is the estimated end-to-end throughput, and $r_l = t_{opt} * M_l$ is the bandwidth to be allocated on each physical link $l \in L$.

Algorithm 1 Compute Resources on Each Link

```
1:  input variables:
2:  $B \leftarrow [b_0, \ldots, b_{N-1}]$ //List of all nodes in the REDiP path
3:  $F$ // Fidelities ($F(b_i, b_{i+1})$ fidelity of segments between $b_i, b_{i+1}$)
4:  $R$ //REDiP rank vector ($R(b_i) \equiv$ rank of $b_i$)
5:  $P$ //REDiP purification ($P(r) \equiv$ # purification rounds at rank $r$)
6:  $H$ //Noise parameters ($H(b_i) \equiv$ parameters of $b_i$, namely $\eta$ and $p_2$)
7:  output variables:
8:  //Counts the resources on each link
9:  $M \leftarrow [1 \forall$ physical link $l$ ]
10: //Estimated end-to-end fidelity
11: $f_{e2e} \leftarrow 0$
12: procedure GETRESOURCES($B, F, R, P, H, M$)
13:     //Purify on lowest rank nodes
14:     $r \leftarrow \min_B \{R(b_i)\}$
15:     $F' \leftarrow \emptyset, B' \leftarrow \emptyset, Z \leftarrow \emptyset$, //Auxiliary variables
16:     for each pair $b_i, b_{i+1} \in B$ do
17:         if $R(b_i) = r$ or $R(b_{i+1}) = r$ then
18:             $m \leftarrow ResourcesDEJMPS(F(b_i, b_{i+1}), P(r), H)$
19:             $f_D \leftarrow FidelityDEJMPS(F(b_i, b_{i+1}), P(r), H)$
20:             $F(b_i, b_{i+1}) \leftarrow f_D$ //Update segment fidelity
21:             for each physical link $l$ on which $(b_i, b_{i+1})$ insist do
22:                 //Update resources on "insisted" physical link
23:                 $M_l \leftarrow M_l * m$
24:             end for
25:         end if
26:     end for
27:     //Exit condition: only end nodes remain (highest rank)
28:     if $length(B) = 2$ then
29:         $f_{e2e} \leftarrow F(b_0, b_1)$
30:         exit
31:     end if
32:     //And Finally Swap
33:     $B' \leftarrow B \setminus \{b_i \in B : R(b_i) = r\}$
34:     for each pair $b'_i, b'_{i+1} \in B'$ do
35:         // $b'_i, b'_{i+1}$ may not be consecutive in $B$...
36:         //...all nodes in $B$ between them must swap
37:         $Z \leftarrow \{b_j \in B : b_j$ between $b'_i$ and $b'_{i+1}\}$
38:         $f_S \leftarrow FidelitySwap(Z, F, H)$
39:         $F'(b'_i, b'_{i+1}) \leftarrow f_S$ //Set the fidelity of this segment
40:     end for
41:     //Recursive call on a reduced set of nodes
42:     $GetResources(B', F', R, P, H, M)$
43: end procedure
```

In conclusion, the REDiP establishment phase, among its other tasks, has the goal of distributing the input variables of Alg. 1 to all REDiP nodes, so that they have the tools to derive the bandwidth to be allocated on every link.

V. GUIDELINES FOR ASSIGNING RANKS

The assignment of REDiP rank and purification vectors are left to the control plane of the quantum network, which has visibility over: (i) the whole network status, (ii) the current set of user requests and their fidelity requirements. In this section, we present a set of guidelines to assign REDiP ranks and purification rounds with the aim of attaining a

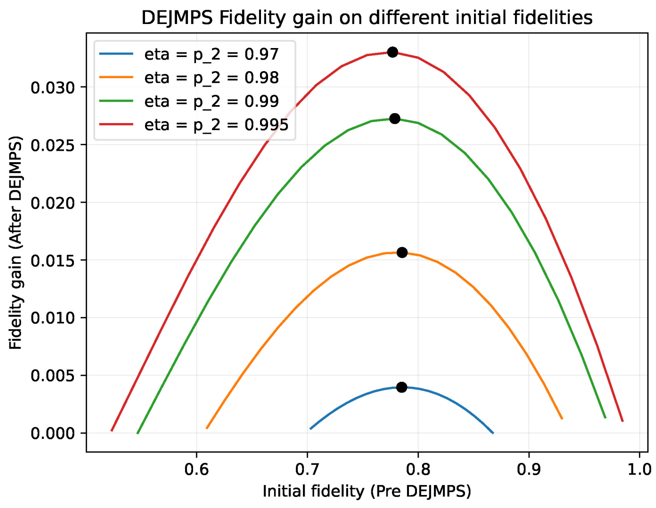

FIGURE 8. **Fidelity gain of DEJMPS purification, i.e., the difference between output and input fidelities. Each curve is relative to a different value of device noise parameters η, p_2, representing measurement and gate success probabilities. Every curve is bound between its F_{min} and F_{max}. Black dots highlight the initial fidelity for which DEJMPS grants the highest fidelity gain.**

target fidelity F_T while maximizing throughput in end-to-end connections. The guidelines underscore the key role of REDiP's flexibility in customizing a PESS to meet user requirements while minimizing the consumed resources.

DEJMPS purification is the tool that REDiP uses to recover the fidelity of entangled links, segments, and connections. However, purification results in a fidelity increase only if the input fidelity falls within the interval (F_{min}, F_{max}), with $F_{min} > 0.5$ and $F_{max} < 1$. The specific values of F_{min} and F_{max} are contingent upon the noise parameters of the involved devices (namely p_2, η). The interested reader can find all the details about these thresholds in [13].

In his seminal work [13], Dür also proposes a PESS tuned on the values of F_{min}, F_{max}. His PESS comprises a sequence of purification and swapping steps. During this process, entangled segments are constantly purified until they reach a "working fidelity" F_W [13] (assumed lower than F_{max}). Subsequently, they are swapped by as many repeaters as possible, ensuring that the fidelity remains higher than F_{min}. The problem with this strategy is that DEJMPS purification performs poorly on segments with a fidelity close to F_{min}: in many cases, it may be convenient to decrease the number of swapping nodes to keep the fidelity higher, even at the cost of an additional purify and swapping step (i.e., an additional rank in REDiP notation).

In Fig. 8, we emphasize this concept by showcasing the *fidelity gain*, defined as the difference between output and input fidelities of DEJMPS purification. The noise parameters η and p_2 are set equal as done in [13], and justified by the noise specifications of current quantum devices like the ones in the IBM cloud.[1] We observe that DEJMPS operates at its best when the input fidelity

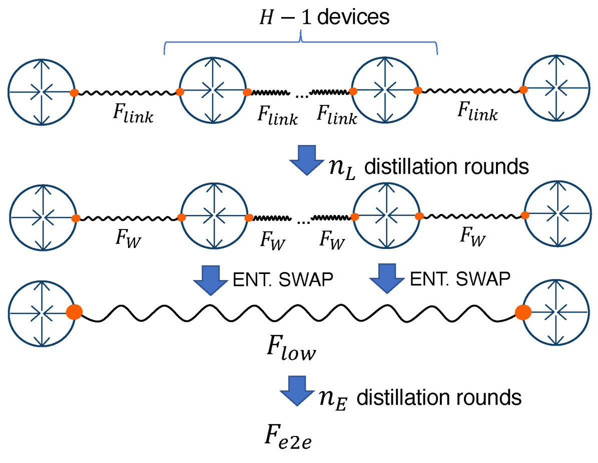

FIGURE 9. **A setup where n_L rounds of purification are carried out at the link level, then all intermediate nodes swap concurrently, and then n_E purification rounds are carried out at the end-to-end level.**

approaches $F_{opt} \approx 0.78$. We can readily observe that the fidelity gain achieved by DEJMPS peaks around this value for all the curves depicted in the plot, as indicated by the black dots. This is what makes our guidelines conceptually independent of the noise parameters η, p_2: the idea is keeping the fidelity equal or higher than F_{opt} so that even a few purification rounds yield a substantial reward.

A. SINGLE SWAPPING STEP

For simplicity, we assume that all devices in the REDiP path have the same noise parameters η, p_2, and all entangled links have the same fidelity F_{link}. We expect that these assumptions will be met by first-generation quantum networks, where very likely all devices and connections share the same technology. We first consider the case where the number of hops H on the path is relatively[2] low, such that all intermediate nodes can concurrently swap without the fidelity dropping too close to F_{min}. We show this scenario in Fig. 9: in REDiP notation, we can implement the PESS shown in the picture as a Parallel ESS. The rank vector is $R = [1, 0, \ldots, 0, 1]$, and the purification vector is $P = [n_L, n_E]$. We named the intermediate fidelities reached after the first purification and swapping steps F_W (working fidelity) and F_{low} respectively. The final fidelity F_{e2e} must be higher than or equal to the target fidelity F_T specified by the user. For any H, we can derive the optimal values for n_L and n_E by determining which combination reaches $F_{e2e} \geq F_T$ by consuming the lowest amount of resources M (here equal for all links), which can be computed using Alg. 1.

B. MULTIPLE SWAPPING STEPS

With longer repeater paths, we might not be able to resolve entanglement distribution in a single swapping step. The idea is to operate recursively: we partition the path into two equal-length sub-paths and apply a nested single swapping

1. https://quantum-computing.ibm.com

2. relatively to the noise parameters

TABLE 1. Rank and purification (R, P) vectors on different REDiP paths. Grey rows highlight the R, P configuration obtained through our guidelines. The link and target fidelities are set to $F_{link} = F_T = 0.95$. The minimum fidelity field represents the minimum fidelity value reached by any pair along the path after a swapping step. The resources per link field indicates the average resources consumed by any physical link on the REDiP path (the maximum value was picked in case of imbalances).

Nodes	η, p_2	Rank Vector	Purify Vector	E2E Fidelity	Resources per Link	Minimum Fidelity
5	0.98	$[2,0,1,0,2]$	$[0,3,3]$	0.9521	239.91	0.8543
		$[1,0,0,0,1]$	$[3,5]$	0.9512	1572.83	0.7467
	0.99	$[2,0,1,0,2]$	$[2,1,1]$	0.9542	26.82	0.9082
		$[1,0,0,0,1]$	$[3,2]$	0.9595	61.70	0.8654
	0.995	$[2,0,1,0,2]$	$[2,1,0]$	0.9559	10.19	0.9559
		$[1,0,0,0,1]$	$[2,1]$	0.9650	10.99	0.9204
	1	$[1,0,0,0,1]$	$[2,0]$	0.9901	4.58	0.9901
7	0.98	$[2,0,0,1,0,0,2]$	$[0,3,4]$	0.9504	937.12	0.7756
		$[1,0,0,0,0,0,1]$	$[2,8]$	0.9503	26212.95	0.6359
	0.99	$[2,0,0,1,0,0,2]$	$[0,3,1]$	0.9520	38.71	0.8164
		$[1,0,0,0,0,0,1]$	$[2,3]$	0.9586	86.02	0.7854
	0.995	$[2,0,0,1,0,0,2]$	$[1,1,1]$	0.9503	13.48	0.8672
		$[1,0,0,0,0,0,1]$	$[3,1]$	0.9514	23.78	0.8926
	1	$[1,0,0,0,0,0,1]$	$[2,0]$	0.9851	4.58	0.9851
9	0.98	$[3,0,1,0,2,0,1,0,3]$	$[0,2,2,4]$	0.9505	2099.36	0.7949
		$[2,0,0,0,1,0,0,0,2]$	$[2,4,4]$	0.9559	12285.19	0.7422
	0.99	$[3,0,1,0,2,0,1,0,3]$	$[0,3,1,1]$	0.9522	71.55	0.8796
		$[2,0,0,0,1,0,0,0,2]$	$[1,3,1]$	0.9502	92.66	0.7822
	0.995	$[3,0,1,0,2,0,1,0,3]$	$[2,1,1,0]$	0.9530	22.32	0.9530
		$[2,0,0,0,1,0,0,0,2]$	$[2,2,0]$	0.9580	23.69	0.9204
	1	$[1,0,0,0,0,0,0,0,1]$	$[2,0]$	0.9803	4.58	0.9803
11	0.98	$[3,0,0,1,0,2,0,0,1,0,3]$	$[0,3,2,4]$	0.9516	6493.34	0.7756
		$[2,0,0,0,0,1,0,0,0,0,2]$	$[2,5,4]$	0.9529	46409.10	0.6857
	0.99	$[3,0,0,1,0,2,0,0,1,0,3]$	$[0,2,2,1]$	0.9527	98.32	0.8164
		$[2,0,0,0,0,1,0,0,0,0,2]$	$[1,3,2]$	0.9588	279.30	0.7359
	0.995	$[3,0,0,1,0,2,0,0,1,0,3]$	$[2,1,0,1]$	0.9513	26.07	0.8869
		$[2,0,0,0,0,1,0,0,0,0,2]$	$[2,1,1]$	0.9554	27.40	0.8988
	1	$[1,0,0,0,0,0,0,0,0,0,1]$	$[2,0]$	0.9754	4.58	0.9754

step on every sub-path independently. Finally, we stitch together the sub-paths by applying a second single swapping step on the central repeater connecting the sub-paths. This recursion can even be pushed forward, as sub-paths can be further partitioned into sub-sub-paths, and so on. We can express this entanglement swapping strategy as a *hybrid* between the Nested and Parallel strategies, which can easily be implemented in REDiP. As an example, if we partition the path only once, the rank vector can be set with the shape

$$R = [2, 0, \ldots, 0, 1, 0 \ldots, 0, 2]. \tag{9}$$

The generalization to include additional nested partitions in R is straightforward. The purification vector P has r elements, where r, the highest rank, is equal to the number of nested partitions plus one.

Now that we fixed the structure of the adopted PESS, we want to determine the optimal configuration, given a REDiP path, and a target end-to-end fidelity F_T. Differently from the single swapping step case, now we have two degrees of freedom: (i) how many nested partitions, and (ii) how many purification rounds to perform at each nested step. In REDiP terms, this translates into determining the number of ranks inside R, and the purification vector P. Even though the domain of this problem is larger than the single swapping step case, the total number of possible combinations remains relatively low, so that the optimal solution can be efficiently found through a full exploration of the domain.

We implemented these guidelines and found the optimal R, P combination on a set of REDiP paths. Table 1 shows our results on twelve representative paths. From these results, we point out some interesting insights.

- First of all, we notice that the *Resources per link*, computed using Alg. 1, are very high when noise parameters η and p_2 are set to 0.98. This highlights the fact that noisy hardware has a destructive effect on the performances of quantum networks deployed within the Noisy-Intermediate Scale Quantum (NISQ) [8] era of quantum computing.

- Second, we can appreciate that optimal *Purification Vectors* (grey rows) apply at least one round of purification on the link level (i.e., at rank zero) only when η and p_2 are equal to 0.995 or 1. This can easily be explained by the fact that DEJMPS purification on the link level is applied to the input fidelity $F_{link} = 0.95$. This value is above or very close to F_{max} when the hardware is noisier, making DEJMPS unrewarding.
- Third, we observe that in the absence of noise ($\eta, p_2 = 1$), it is always optimal to purify on the link level to reach a very high fidelity, and then swap on all intermediate nodes in a single round.
- Fourth, we tracked the fidelity after each swapping step and reported the minimum value for each path in the *Minimum Fidelity* column. We can appreciate that the optimal configurations (grey rows) respect the observation we derived from Fig. 8: fidelity never drops far below $F_{opt} \approx 0.78$.
- Finally, guidelines can guarantee a significant reduction of consumed resources. Numerical evidence of this statement is shown in Table 1, where we appreciate the difference in performances between the optimal configurations (grey rows) and other representative configurations (white rows) under identical physical parameters η, p_2. As expected, the guidelines prove particularly effective for near-term noise parameters ($\eta, p_2 \leq 0.995$). For example, we see that the estimated resources consumed on each link to reach the target end-to-end fidelity decreases by a factor that even surpasses 5 when $\eta, p_2 = 0.98$.

Our findings indicate that both rank and purification vectors play a crucial role in maintaining resource efficiency. They effectively prevent a significant drop in fidelity following entanglement swapping steps, thereby enabling subsequent DEJMPS processes to swiftly restore high fidelity values within just a few rounds. This strategy minimizes resource consumption while maximizing fidelity restoration during the entanglement manipulation.

VI. EXPERIMENTS

We implemented a REDiP simulator to estimate the impact of several PESSs on the protocol performances in terms of throughput (*pairs/s*) and fidelity. The simulator was implemented as a Python package on top of Netsquid [24], a simulation engine for quantum networks. Below we define the assumptions of the simulation campaign and then we show some representative scenarios. From the simulation outcomes, it is clear that the assignments of REDiP ranks and purification rounds have a significant impact on the entanglement distribution performance, confirming the estimations of Sections IV and V.

A. ASSUMPTIONS

In our simulation scenarios we assume all repeaters and links have the same hardware specifications. In particular, we modeled an implementation of the Midpoint Source (MS for short) Link layer protocol defined in [25], and we set its parameters according to the *optimistic* parameter set used in [26]. The rationale behind this choice is that [26] employs the most advanced emerging technology for multi-mode quantum memories.

To model quantum errors, we introduced three sources of noise: (i) An initial depolarization probability p_0^d applied on both endpoints of a newly generated link, so that its initial fidelity is $F_{link} < 1$. (ii) A quantum memory exponential dephasing rate r_d, which simulates the decoherence of a qubit state $\hat{\rho}$ stored inside a quantum memory, so that after Δt time the new state $\hat{\rho}'$ of the qubit can be written as

$$\hat{\rho}' = f(\hat{\rho}, \Delta t) \; = e^{-r_d \Delta t}\hat{\rho} + \left(1 - e^{-r_d \Delta t}\right) Z\hat{\rho}Z. \qquad (10)$$

We also define the coherence time T_c of a quantum memory as the time after which the dephasing probability reaches 5%. Formally,

$$T_c = \frac{-\log 0.95}{r_d}. \qquad (11)$$

(iii) A depolarization probability p_{err}^d applied right before each measurement and 2-qubit quantum gate, to introduce a noise component in entanglement swapping and purification. We point out that since depolarization is a Pauli noise, then p_{err}^d is bound to the accuracy p_2 of 2-qubit gates by the following relationship [27], [28]:

$$(1 - p_2) = \frac{5}{2} p_{err}^d. \qquad (12)$$

All physical links between the devices were modeled as $15km$ long fiber-optic links. We employed the DEJMPS protocol whenever entanglement purification was required.

B. GUIDELINES VERIFICATION

First of all, we employed simulations to validate the estimations from Table 1. As anticipated, we dropped the two assumptions that guarantee the perfect accuracy of the estimations. This setup was reached by using only free quantum memory slots to generate new entanglement, and by introducing a quantum memory coherence time $T_c = 5ms$, which is in line with near-term hardware specifications [22], [26].

We show our results in Table 2. As we could expect, the estimated results of end-to-end fidelity are upper bounds to the end-to-end fidelity reached in these simulated scenarios. Moreover, the estimated number of consumed resources is a lower bound to the values obtained in the simulations. This is due to two main factors: (i) the dephasing noise introduced by the realistic quantum memory reduces both the fidelity and the probability of success of every DEJMPS round. Moreover, (ii) some entangled qubits reach their cutoff time of $5ms$ and are automatically released to the Link

TABLE 2. **Simulated results of the set of PESSs shown in Table 1, obtained from the guidelines. We show the actual resources per link and end-to-end fidelity on a realistic scenario with $T_c = 5ms$. For simplicity, we also report in red the estimated values from Table 1.**

Nodes	η, p_2	Resources per Link	E2E Fidelity
5	0.99	50.56 (26.82)	0.951 (0.954)
	0.995	13.42 (10.19)	0.953 (0.956)
	1	8.13 (4.58)	0.969 (0.990)
7	0.99	305.67 (38.71)	0.941 (0.959)
	0.995	23.87 (13.48)	0.925 (0.951)
	1	8.13 (4.58)	0.952 (0.980)
9	0.995	54.12 (22.32)	0.910 (0.953)
	1	8.13 (4.58)	0.941 (0.980)
11	0.995	124.37 (26.07)	0.902 (0.951)
	1	8.13 (4.58)	0.927 (0.975)

layer for new entanglement generation. These qubits are considered wasted resources, increasing the total number of consumed resources. The impact of cutoff qubits grows with the number of purification rounds and with the number of nodes on the path. As an example, we take the 9- and 11-node paths with $\eta, p_2 = 0.995$ (seventh and ninth rows of Table 2): the resources per link are more than double for the 11-node path because the path is longer and purification is performed at a higher rank, inducing a larger delay. These two factors cause a large fraction of qubits stored in the quantum memory to reach their cutoff time. The scenarios that are missing in Table 2 are those where the coherence time is not long enough, and we are not able to deliver end-to-end entanglement to applications. From these results, we conclude that our estimations are much closer to the experimental result if the fraction of cutoff qubits is negligible with respect to the total consumed resources. We point out that these PESSs obtained through our guidelines are confirmed to be good choices by the simulations. We tested all other PESSs obtainable according to the scheme from Section V-B, and we verified that even though the estimation accuracy varies depending on the scenario, the guidelines still get the PESS that consumes the least amount of resources.

C. SIMULATION RESULTS

We also carried out a set of simulated experiments on a representative network topology composed of 9 nodes connected as a linear chain. We set $p_0^d = 0.01334$ so that the initial fidelity of link-generated pairs is approximately $F_{link} = 0.98$. We analyze the performances of different PESSs, all implemented with REDiP, under varying quantum noise.

Fig. 10 reports the results of our simulations. Specifically, Figs. 10(a) and 10(b) plot the end-to-end throughput (*pairs/s*) and average (squared) fidelity as a function of coherence time

TABLE 3. **PESS specifications used within the simulated experiments.**

PESS	R	P
Parallel	$[1, 0, \ldots, 0, 1]$	$[0, 0]$
Nested	$[3, 0, 1, 0, 2, 0, 1, 0, 3]$	$[0, 0, 0, 0]$
*Nested**	”	$[0, 0, 1, 0]$
*Nested***	”	$[0, 1, 1, 0]$
Consecutive	$[7, 0, 1, 2, 3, 4, 5, 6, 7]$	$[0, \ldots, 0]$
*Consecutive**	”	$[0, 0, 1, 0, \ldots, 0]$
*Consecutive***	”	$[0, 0, 1, 0, 1, 0, 0, 0]$
*Hybrid**	$[2, 0, 1, 0, 1, 0, 1, 0, 2]$	$[0, 1, 0]$

T_c, keeping $p_{err}^d = 0.005$ as a constant, whereas Figs. 10(c) and 10(d) plot the same metrics as a function of p_{err}^d, keeping $T_c = 5ms$ as a constant. Such constant values were picked in line with near-term hardware specifications [22], [26]. In all these plots, we compare the performances of a set of eight representative PESSs. For the sake of readability, we named each PESS by the name of the entanglement swapping strategy and postponed a number of stars (“$*$”) equal to the total rounds of purification applied. To avoid ambiguities, we provide in Table 3 the vectors R and P used on each PESS (we do not repeat R across strategies sharing the same ESS). Regarding the *Hybrid** PESS, it has been added to highlight how REDiP flexibility allows to mix up different strategies: 0-ranked nodes are assigned as in the Nested ESS, but all remaining repeaters are 1-ranked, so that they swap in parallel. This behavior makes this strategy a hybrid between Nested and Parallel.

We can see from Figs. 10(a) and 10(c) that the throughput highly depends on the total number of purification rounds and that purification performed at higher ranks costs slightly more in terms of throughput due to the longer transmission time of purification control messages. REDiP can be set up by the user with a suitable PESS that aims to maximize the throughput while delivering entanglement with a guaranteed minimum fidelity. For example, *Parallel* is a good option for low fidelity requirements, especially when T_c is critically low. If instead, requirements impose a very high fidelity, then the best option is *Nested***, which outperforms all other PESSs involved in these simulations (Figs. 10(b) and 10(d)), at the cost of a lower throughput. A good trade-off between the two is the *Hybrid** PESS, which exploits part of the benefits from both strategies: it has indeed a higher throughput than *Nested** and a comparable fidelity. As a general consideration emerging from these results, we can say that purification combined with the Consecutive ESS, given its intrinsic asymmetry, does not provide good performances with respect to other strategies. Finally, we see from Fig. 10(a) that some strategies with one and two degrees of purification are not able to deliver entanglement if the coherence time T_c is below a certain threshold. This happens because qubits are automatically released by a *cutoff* mechanism if they stay in a quantum memory for a time higher than T_c.

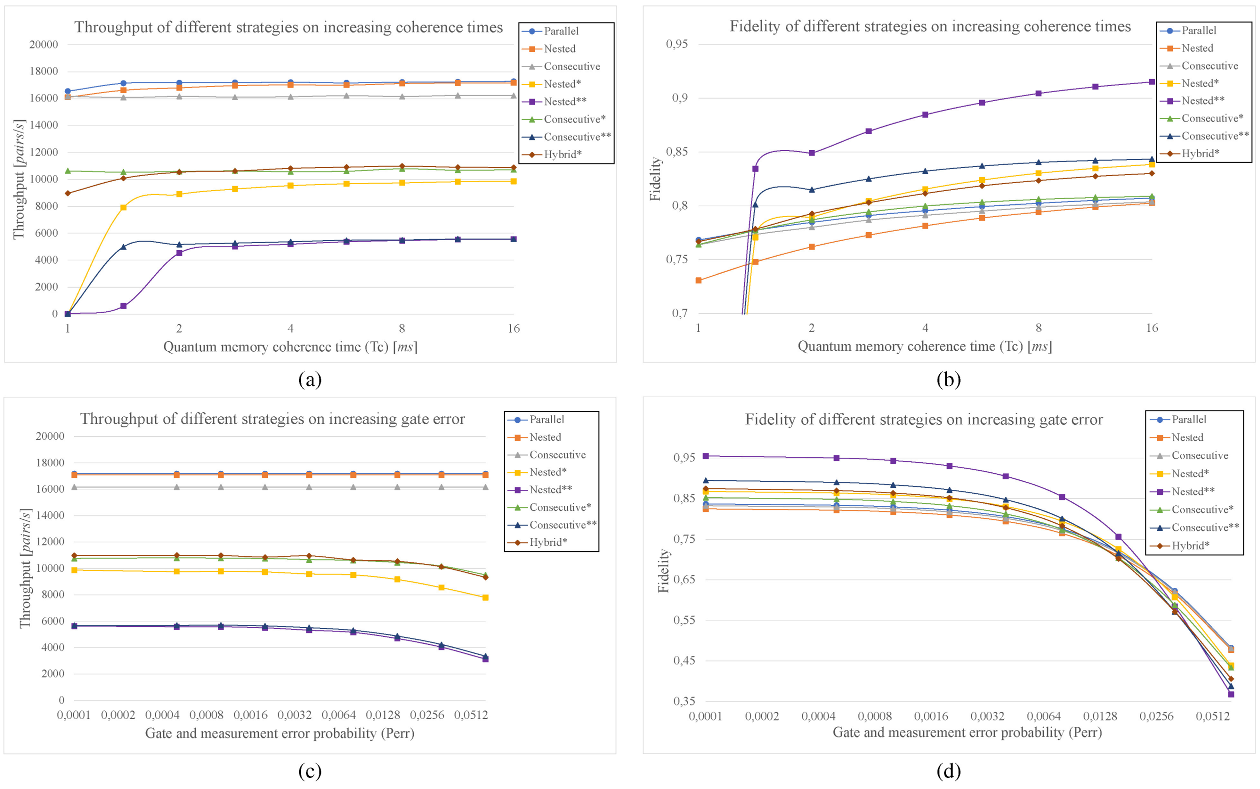

FIGURE 10. Results of the simulation campaign on a nine nodes chain, showing (a) throughput, (b) fidelity as a function of memory coherence time T_c, and (c) throughput, (d) fidelity as a function of gate error p^d_{err}. Each marker was obtained as the average of 31 independent runs. Confidence intervals are not shown as their size is smaller or comparable to the markers.

VII. CONCLUSION

REDiP is a configurable protocol for entanglement distribution in quantum networks. It exposes a flexible service to the upper layer, which is able to set up a custom Purify-and-Swap chain by means of a single protocol instance. REDiP design relies on implementing the intuitive mechanism of "ranks", that define a timing hierarchy among all REDiP nodes.

In this work, we also provided an algorithm to estimate the target fidelity of end-to-end connections obtained with REDiP, and the amount of resources consumed on each physical link for each end-to-end connection. We leveraged this algorithm and the properties of DEJMPS purification to delineate guidelines for the assignment of REDiP ranks that achieve a target fidelity with a reasonable amount of resources. We brought numerical evidence that a correct assignment of ranks leads in many cases to a massive saving of resources, that may even surpass a factor of 5 with noisy hardware.

Our simulated experiments validated our observations and showed that in a real-world scenario, the selection of an appropriate swapping and purification strategy is of even greater importance. For instance, implementing the Nested strategy resulted in a 5 percent increase in end-to-end fidelity compared to an asymmetric choice such as the Consecutive strategy, while maintaining a similar throughput. Results suggested a line of further research in the analysis of quantum memory occupancy of REDiP nodes. Such a study would potentially refine the estimations of our algorithm to include the impact of the average memory occupancy and decoherence of each device on REDiP performances.

As research on the Quantum Internet continues to develop, protocols like REDiP play a pivotal role in addressing the intricate data plane requirements. By thoroughly investigating these issues and enabling fine-tuning solutions, REDiP guarantees the flexibility required to adapt to evolving quantum technologies and standards.

REFERENCES

[1] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello, "Distributed quantum computation over noisy channels," *Phys. Rev. A*, vol. 59, no. 6, pp. 4249–4254, Jun. 1999. [Online]. Available: https://doi.org/10.1103/physreva.59.4249

[2] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal blind quantum computation," in *Proc. 50th Annu. IEEE Symp. Found. Comput. Sci.*, 2009, pp. 517–526.

[3] A. K. Ekert, "Quantum cryptography based on bell's theorem," *Phys. Rev. Lett.*, vol. 67, pp. 661–663, Aug. 1991. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.67.661

[4] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theor. Comput. Sci.*, vol. 560, pp. 7–11, Dec. 2014. [Online]. Available: https://doi.org/10.1016/j.tcs.2014.05.025

[5] M. Walter, D. Gross, and J. Eisert, *Multipartite Entanglement*. Hoboken, NJ, USA: Wiley, 2016, pp. 293–330. [Online]. Available: http://arxiv.org/abs/1612.02437

[6] S. Muralidharan, L. Li, J. Kim, N. Lutkenhaus, M. Lukin, and L. Jiang, "Optimal architectures for long distance quantum communication," *Nature*, Jun. 2016.

[7] L. Jiang, J. M. Taylor, K. Nemoto, W. J. Munro, R. Van Meter, and M. D. Lukin, "Quantum repeater with encoding," *Phys. Rev. A*, vol. 79, Mar. 2009, Art. no. 032325. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.79.032325

[8] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: https://doi.org/10.22331%2Fq-2018-08-06-79

[9] J. Illiano, M. Caleffi, A. Manzalini, and A. S. Cacciapuoti, "Quantum Internet protocol stack: A comprehensive survey," *Comput. Netw.*, vol. 213, Aug. 2022, Art. no. 109092. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622002250

[10] A. Dahlberg et al., "A link layer protocol for quantum networks," in *Proc. ACM Spec. Interest Group Data Commun.*, 2019, pp. 159–173. [Online]. Available: https://doi.org/10.1145/3341302.3342070

[11] R. Van. Meter et al., "A quantum internet architecture," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, 2022, pp. 341–352. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/QCE53715.2022.00055

[12] A. Pirker and W. Dür, "A quantum network stack and protocols for reliable entanglement-based networks," *New J. Phys.*, vol. 21, no. 3, Mar. 2019, Art. no. 033003. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2Fab05f7

[13] W. Dür, H.-J. Briegel, J. I. Cirac, and P. Zoller, "Quantum repeaters based on entanglement purification," *Phys. Rev. A*, vol. 59, pp. 169–181, Jan. 1999. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.59.169

[14] L. Bacciottini, L. Lenzini, E. Mingozzi, and G. Anastasi, "A configurable protocol for quantum entanglement distribution to end nodes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2023, pp. 3485–3490.

[15] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Phys. Rev. Lett.*, vol. 77, pp. 2818–2821, Sep. 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.77.2818

[16] C. Cicconetti, M. Conti, and A. Passarella, "Request scheduling in quantum networks," *IEEE Trans. Quantum Eng.*, vol. 2, pp. 2–17, Jun. 2021.

[17] M. Pant et al., "Routing entanglement in the quantum Internet," *npj Quantum Inf.*, vol. 5, p. 25, Dec. 2019.

[18] S. Pouryousef, N. K. Panigrahy, and D. Towsley, "A quantum overlay network for efficient entanglement distribution," 2023, *arXiv:2212.01694*.

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

[20] J. Li, Q. Jia, K. Xue, D. S. L. Wei, and N. Yu, "A connection-oriented entanglement distribution design in quantum networks," *IEEE Trans. Quantum Eng.*, vol. 3, pp. 1–13, May 2022.

[21] L. Aparicio, R. Van Meter, and H. Esaki, "Protocol design for quantum repeater networks," in *Proc. 7th Asian Internet Eng. Conf.*, 2011, pp. 73–80. [Online]. Available: https://doi.org/10.1145/2089016.2089029

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

[23] R. van Meter, J. Touch, and C. Horsman, "Recursive quantum repeater networks," *Prog. Inform.*, pp. 65–79, Mar. 2011. [Online]. Available: https://www.nii.ac.jp/pi/

[24] T. Coopmans et al., "NetSquid, a NETwork simulator for QUantum information using discrete events," *Commun. Phys.*, vol. 4, no. 1, p. 164, Jul. 2021. [Online]. Available: https://doi.org/10.1038/s42005-021-00647-8

[25] C. Jones, D. Kim, M. T. Rakher, P. G. Kwiat, and T. D. Ladd, "Design and analysis of communication protocols for quantum repeater networks," *New J. Phys.*, vol. 18, no. 8, Aug. 2016, Art. no. 083015. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2F18%2F8%2F083015

[26] D. Yoshida, K. Niizeki, S. Tamura, and T. Horikiri, "Entanglement distribution between quantum repeater nodes with an absorptive type memory," *Int. J. Quantum Inf.*, vol. 18, no. 5, Aug. 2020, Art. no. 2050026. [Online]. Available: https://doi.org/10.1142%2Fs0219749920500264

[27] Y. R. Sanders, J. J. Wallman, and B. C. Sanders, "Bounding quantum gate error rate based on reported average fidelity," *New J. Phys.*, vol. 18, no. 1, Dec. 2015, Art. no. 012002. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2F18%2F1%2F012002

[28] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Phys. Rev. A*, vol. 85, Apr. 2012, Art. no. 042311. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.85.042311

Open Access funding provided by 'Università degli Studi di Firenze' within the CRUI CARE Agreement