Online Transit Entanglement Routing in Quantum Networks: Architecture Design and Optimization

Wan-Ting Ho[§], Shing-Yan Fang[§], Wei-Chia Hsieh[‖†], Li-Feng Chen[‖†], Jian-Jhih Kuo[*†], and Ming-Jer Tsai[§]
[§]Dept. of Computer Science, National Tsing Hua University, Hsinchu Taiwan
[†]Dept. of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan

***Abstract*—Quantum networks (QNs) gradually gain significant attention due to their higher security compared with classical networks. Conventional approaches in QN routing often aggregate multiple requests into a batch before determining their routing paths. However, this approach may overlook the limited lifetime of qubits, resulting in critical decoherence. In this paper, we present a new online entanglement routing architecture with an online optimization problem and propose a novel $[1, O(\log |V|)]$-competitive algorithm supporting online requests with admission control, aiming to maximize the number of admitted requests. Finally, extensive simulation results show that our algorithm can outperform the existing approaches by up to 98%.**

I. Introduction

Quantum networks (QNs) have been proposed to transmit quantum bits (i.e., qubits) between quantum nodes [1]. Compared to classical networks, QNs can facilitate 1) quantum key distribution (QKD), which generates and distributes a random secret key known only to the two parties for encrypting and decrypting messages [2]–[4]. 2) distributed quantum computing, which creates more computing potential by linking multiple physically separated quantum computers [1], [5], [6]. Generally speaking, a QN consists of quantum nodes, each of which has a specific amount of quantum memory to store qubits and mitigate rapid decoherence. Any pair of adjacent nodes are interconnected through quantum channels, and they can establish an *entangled link* over their quantum channel before transmitting a data qubit, as shown in Fig. 1(a). Besides, each entangled link can transmit only one data qubit [7], [8], and entangled links will be destroyed after transmission. Moreover, each qubit has a limited lifetime, so data qubits and entangled links remain active and effective for a finite period [9].

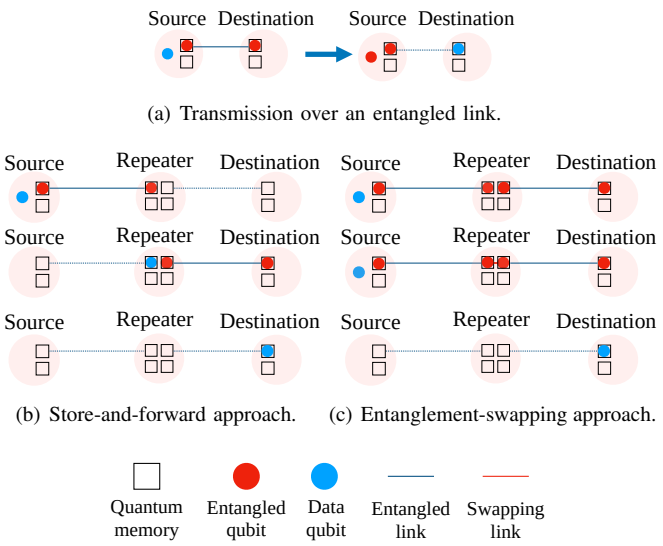
Nonetheless, creating long-distance entangled links between non-adjacent nodes remains challenging [7]. To this end, two transmission approaches have been discussed in the literature. *1) Store-and-forward approach* [9]. The data qubit is iteratively stored in a repeater [10], awaiting forwarding until the entangled link for the next hop is established, as shown in Fig. 1(b). This strategy breaks up the long-distance path from source to destination into small segments, which helps increase the success transmission probability. However, the store-and-forward method poses potential risks, since the repeaters handle the data. *2) Entanglement-swapping approach* [8], [11]. It first creates entangled links between adjacent nodes along the path from the source to the destination, called external links [11], [12]. Subsequently, repeaters perform entanglement swapping, logically creating *swapping links*, called internal links, within repeaters [11], [12]. These internal links logically patch together the entangled links incident on them to form an entangled path, which is essentially an end-to-end entangled pair. This way allows the source to transmit data to the destination securely without the need for temporary storage in repeaters. Nevertheless, both entangling and swapping processes carry the unsuccessful risk, and a complete entangled path is needed before transmission, leading to a longer transmission time than the store-and-forward approach [9]. Thus, it is interesting and important to combine the advantages of the two approaches to develop an advanced routing architecture.

Moreover, the existing routing architectures and algorithms for QNs in the literature have at least one of the following issues. 1) *Online requests are not supported*. Most algorithms (e.g., [12]–[14]) work in an offline manner, requiring the QN controller in the routing architecture to gather multiple requests *in a batch* and then select entangled paths for these requests. Nonetheless, batch processing may neglect individual data qubits' limited lifetime and idle them, causing crucial decoherence. 2) *Congestion happens frequently*. Many algorithms (e.g., [8], [11]) exploit shortest paths without considering load balancing, tending to generate hot spots in QNs and resulting in serious congestion and inefficient network utilization. The issues are exacerbated especially as quantum memory is usually scarce in QNs [14]. 3) *Multi-commodity flow based methods are time-consuming*. To achieve the maximum throughput, it is intuitive to adopt the techniques for solving the multi-commodity integral flow problem (e.g., [12]–[14]). Specifically, the entangling and swapping processes are typically modeled as two or more integer linear programmings (ILPs), each of which is relaxed as an linear programming (LP) and solved by an LP solver. Then, the integral solution is obtained by rounding the LP solutions. This way usually yields satisfactory routing paths but fails to provide real-time responses to requests since solving LPs is computationally intensive, making data qubits and entangled pairs idle and decohere over time. Thus, a fast online admission control and routing algorithm is necessary.

In light of the aforementioned challenges, we present a novel **O**nline **T**ransit **E**ntanglement **R**outing (OTER) architecture. To support online requests, when a new request arrives, the OTER immediately determines whether to admit it and allocate the resources for it. In addition, the OTER uses both the store-

* indicates the corresponding author; ‖ denotes the equal contributions.
{s111062573, s11106553}@m111.nthu.edu.tw, {swjia111u, clfeng111u, lajacky}@cs.ccu.edu.tw, mjtsai@cs.nthu.edu.tw

This work was supported by the National Science and Technology Council, Taiwan, under Grants 111-2221-E-007-044-MY3, 111-2628-E-194-001-MY3, and 113-2221-E-194-040-MY3.



(a) Transmission over an entangled link.

(b) Store-and-forward approach. (c) Entanglement-swapping approach.

Fig. 1. Different types of transmission in quantum networks.

and-forward and entanglement-swapping approaches to break up every request into two predefined segments by anchoring a transit node. This design can generate more diverse possible routing paths and efficiently mitigate the hot spot issue caused by using shortest paths. Last, the OTER combines predefined segments to derive routing paths instead of exhaustively searching all possible paths. Thus, it suffices to determine the transit node for each newly arriving request so that the OTER can significantly decrease the computational complexity. However, the admitted requests and the positions of their transit nodes must be carefully determined, but these critical factors are not examined by the existing routing algorithms for QNs.

To fully utilize the advantages of the OTER, we make the first attempt to explore the *fast online transit entanglement routing*. Simultaneously determining the admitted requests and their transit nodes raises the following challenges. 1) *Online admission control*. The decision of which requests to accept is a crucial issue in online routing. If accepting a request would occupy significant portion of network resources, leading to subsequent requests being unable to be serviced, then such a request must be carefully considered for rejection. 2) *Trade-off between success probability and transit storage*. Intuitively, anchoring a transit node to break up a routing path into two predefined segments can help increase the success transmission probability since the two segments can be established individually. However, the transit node would consume one more unit of its quantum memory to store the qubit, which might result in higher potential resource competition. 3) *Transit node selection*. Selecting a transit node far from the shortest path tends to make the routing path zigzag and mitigate the hot spot issue. Nonetheless, such a routing path may consume more resources. Therefore, fast online transit entanglement routing is challenging because it needs to jointly decide whether to admit a request and whether / where to anchor a transit node.

To address the above challenges, we formulate a new optimization problem named Online **Q**uantum Routing **A**dmission **C**ontrol and **T**ransit **S**election Problem (Q-ACTS) to maximize the number of admitted requests. We prove that the Q-ACTS does not admit any $(\frac{|V|}{2}-1)$-competitive algorithm, where $|V|$ denotes the number of nodes in the QN. For the Q-ACTS, we design a $[1, O(\log|V|)]$-competitive algorithm, termed **T**ransit **A**nchoring and **S**egment **C**oncatenation Algorithm (TASC). To evaluate the resource efficiency of the routing path anchored at a different trust node, we introduce the notion of *resource efficiency index (REI)* based on the *primal-dual analysis*. The TASC tends to use a path with a higher REI to serve each request. Plus, with the subtly-designed initialization and update rule for dual variables, the TASC can achieve the bicriteria competitive ratio. Due to the page limit, the related works are discussed in Appendix A of [15]. Finally, the simulation results show that the TASC outperforms the existing methods by 98%.

II. System Model and Problem Formulation

In this section, we introduce the QN components and explain how the OTER works. Then, we formulate the offline problem of the Q-ACTS for maximizing the number of served requests.

A. Basic QN Components

A QN is modeled as an undirected graph $G = (V, E)$, where V and E respectively represent the quantum nodes and channels (i.e., edges) connecting each adjacent pair of quantum nodes. A central controller communicates with all nodes to collect entangling and swapping results and manages global information in G. Every quantum $u \in V$ has a limited amount of quantum memory $m(u)$ to store data qubits or entangled pairs for a limited lifetime. Note that the lifetime typically ranges from seconds to minutes according to the material and technology [16]. We follow [9] to divide a lifetime into L time slots, and each time slot includes the time required for conducting one entangling and multiple swapping operations.

The success probability of creating an entangled link between nodes u and v, where $(u, v) \in E$, is represented by $\mathrm{P}_{u,v}$, which decays exponentially with the length of the quantum channel, i.e., $\mathrm{P}_{u,v} = e^{-a \cdot l(u,v)}$, where a is a constant based on the material and technology [8]. Moreover, two entangled links incident to any node $u \in V$ can be glued together by a swapping link generated through conducting the entanglement swapping at u. Conducting the swapping process at each node $u \in V$ also has a success probability, denoted as Q_u. Therefore, the probability of successfully sending a qubit from node i to node j along path $P(i, j)$ can be expressed as:

$$\Pr(P(i,j)) = \prod_{(u,v)\in P(i,j)} \mathrm{P}_{u,v} \prod_{u\in \overline{P}(i,j)} \mathrm{Q}_u, \qquad (1)$$

where $\overline{P}(i, j)$ is the set of nodes performing swapping in path $P(i, j)$, i.e., those in $P(i, j)$ except node i and node j.

B. Online Transit Entanglement Routing (OTER)

Initially, the OTER finds the routing path with the highest success probability[1] for each pair of nodes in the QN as

[1]For ease of presentation, in this paper, the OTER only chooses one path even if there are multiple paths with the same success probability between a pair of nodes. However, in practice, the OTER can be extended painlessly to store top-k paths for each pair of nodes for path diversity, where $k \in \mathbb{N}$.

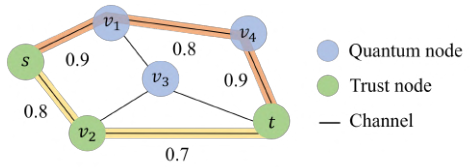


Fig. 2. Example of success probability of path construction, where the value on the channel represents the success probability of entangled.

the predefined segment between them. For clarity, we use $\ell(i,j)$ to denote the predefined segment between node i and node j. Once a request arrives, the OTER examines every candidate transit node and concatenates the corresponding two predefined segments to derive the candidate paths. This design can significantly reduce delay compared to calculating routing paths from scratch. Besides, it allows data qubits to temporarily progress to transit nodes without waiting for the establishment of complete entangled paths and can avoid network hot spots by anchoring appropriate transit nodes [9]. To this end, the OTER considers user trust in quantum nodes when selecting transit nodes for users' requests to prevent data eavesdropping. For ease of presentation, let R denote a set of online requests, and each request $r \in R$ has a set of trusted nodes, denoted by T_r. Note that data qubits can also be sent without using a transit node if the direct transmission is faster and not congested.

Given the success probabilities of entangling and swapping, we analyze the success probability of establishing a path in the OTER for request r through transit node k and not, as follows. Due to the page limit, the proof is left in Appendix B in [15].

Theorem 1 (Success probability of path construction in the OTER)**.** Let $F_1(\tau)$ and $F_2(\tau)$ denote the successful probabilities of path construction with exactly one and two segments at the time slot τ. Within a lifetime L, the successful probabilities of path construction for request r directly (case 1) and via the trusted transit node k (case 2) can be derived by

$$p(r,k) = \begin{cases} \sum_{\tau=1}^{L} F_1(\tau), \text{ (case 1) if } k \in \{s(r), t(r)\}; \\ \sum_{\tau=2}^{L} F_2(\tau), \text{ (case 2) otherwise,} \end{cases} \tag{2}$$

where $s(r)$ and $d(r)$ denote the source and destination of request r, respectively, and

$$\begin{aligned} F_1(\tau) =& \Big(1 - \Pr\big(\ell(s(r),t(r))\big)\Big)^{\tau-1} \Pr\big(\ell(s(r),t(r))\big), \\ F_2(\tau) =& \frac{\Pr\big(\ell(s(r),k)\big) \cdot \Pr\big(\ell(k,t(r)))\big)}{1 - \Pr\big(\ell(s(r),k)\big)} \Big(1 - \Pr\big(\ell(k,t(r)))\big)\Big)^{\tau-1} \\ & \cdot \sum_{m=1}^{\tau-1} \Big(\frac{1 - \Pr\big(\ell(s(r),k)\big)}{1 - \Pr\big(\ell(k,t(r)))\big)}\Big)^{m}. \end{aligned}$$

Fig. 2 shows a calculation example for the success probability of path construction from s to t in the OTER. The orange line represents directly constructing the path for request r (case 1), while the yellow line indicates establishing the routing path through transit node v_2 (case 2). Assume that the success probability of conducting swapping at each node is 1. We can first calculate $\Pr\big(\ell(s,t)\big) = 0.9 \cdot 0.8 \cdot 0.9 \cdot 1 \cdot 1 = 0.648$. Thus, the successfully established OTER paths at time slot τ can be derived as:

$$\begin{aligned} F_1(\tau) &= \Big(1 - \Pr\big(\ell(s,t)\big)\Big)^{\tau-1} \cdot \Pr\big(\ell(s,t)\big) \\ &= (1 - 0.648)^{\tau-1}(0.648), \\ F_2(\tau) &= \frac{\Pr\big(\ell(s,v_2)\big) \cdot \Pr\big(\ell(v_2,t)\big)}{1 - \Pr(\ell(s,v_2))} \Big(1 - \Pr\big(\ell(v_2,t)\big)\Big)^{\tau-1} \\ &\quad \cdot \sum_{m=1}^{\tau-1} \Big(\frac{1 - \Pr\big(\ell(s,v_2)\big)}{1 - \Pr\big(\ell(v_2,t)\big)}\Big)^{m} \\ &= \frac{0.8 \cdot 0.7}{1 - 0.8}(1 - 0.7)^{\tau-1} \sum_{m=1}^{\tau-1} \big(\frac{1 - 0.8}{1 - 0.7}\big)^{m}. \end{aligned}$$

We then analyze the number of generating entangled links and the amount of used quantum memory for a routing path. Let $f_{\ell(i,j)}(u,v)$ denote the number of entangled links which will be generated on link (u,v) if segment $\ell(i,j)$ is used. Specifically, if $\ell(i,j)$ passes through link (u,v), then $f_{\ell(i,j)}(u,v)$ is 1; otherwise, $f_{\ell(i,j)}(u,v)$ is 0. Subsequently, let $g_r^k(u,v)$ represent the number of entangled links needed to be established for a request r on a specific link (u,v) for deriving the routing path from $s(r)$ to $d(r)$. Since such a path consists of one or two predefined segments, similar to Eq. (2), it has two cases:

$$g_r^k(u,v) = \begin{cases} f_{\ell(s(r),t(r))}(u,v) & \text{if } k \in \{s(r), t(r)\}; \\ f_{\ell(s(r),k)}(u,v) + f_{\ell(k,t(r))}(u,v) & \text{otherwise.} \end{cases}$$

Besides, let $h_r^k(u)$ denote the amount of memory consumed at the node u when a request r routes through transit node k. Request r routed via k will consume the memory on node u for 1) establishing links if its routing path visits u and 2) temporarily storing data if k is its transit node and $u = k$, so

$$h_r^k(u) = \sum_{v:(u,v)\in E} g_r^k(u,v) + \begin{cases} 0 & \text{if } k \in \{s(r), t(r)\} \\ & \text{or } u \neq k; \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

C. Problem Formulation – Q-ACTS

The Q-ACTS considers a QN $G = (V, E)$ with a central server, and requests R arrive one by one in an online manner. Each request $r \in R$ has an expected demand size $d(r) \in \mathbb{Z}^+$ (i.e., transmitting $d(r)$ data qubits periodically from its source $s(r)$ to its destination $t(r)$ after arrival). Every request r has its trust nodes set $T_r \subseteq V$, which can serve as its transit node. The Q-ACTS aims to 1) admit a subset of requests, 2) determine whether to use a transit node and select a suitable one, and 3) maximize the total expected demand size of admitted requests. The ILP of the offline Q-ACTS can be written as follows.

$$\max_{y_r, z_r^k} \quad \sum_{r\in R} d(r) \cdot y_r \tag{5a}$$

$$\text{s.t.} \quad \sum_{k\in T_r} p(r,k) \cdot z_r^k \geq d(r) \cdot y_r, \quad \forall r \in R \tag{5b}$$

$$\sum_{r\in R} \sum_{k\in T_r} h_r^k(u) \cdot z_r^k \leq m(u), \quad \forall u \in V, \tag{5c}$$

$$y_r \in \{0,1\}, \quad \forall r \in R, \tag{5d}$$
$$z_r^k \in \{0\} \cup \mathbb{Z}^+, \quad \forall k \in T_r,\ \forall r \in R, \tag{5e}$$

where *decision variables* $y_r \in \{0,1\}$ and $z_r^k \in \{0\} \cup \mathbb{Z}^+$ represent whether request $r \in R$ is admitted and the number of entangled paths required to be established for request $r \in R$ through node $k \in T_r$. Specifically, $y_r = 1$ if and only if request r is admitted. On the other hand, if $z_r^k \geq 1$ and $k \notin \{s(r), t(r)\}$, then request r anchors transit node k and combines the segments $\ell(s(r), k)$ and $\ell(k, t(r))$ as the routing path; if $z_r^k \geq 1$ and $k \in \{s(r), t(r)\}$, then r directly exploits the $\ell(s(r), t(r))$ as the routing path without using a transit node; otherwise, r is not admitted. Specifically, the objective (5a) maximizes the total expected demand size. Constraint (5b) guarantees if request r is admitted, the OTER must allocate sufficient resources to request so that the expected number of transmitted data qubits is at least $d(r)$ for increasing the total expected demand size by $d(r)$; otherwise, no profit will be acquired. Constraint (5c) ensures each node's quantum memory capacity will not be exceeded.

The Q-ACTS has no $(\frac{|V|}{2} - 1)$-competitive algorithm. Due to the page limit, the proof is presented in Appendix C in [15].

Theorem 2. No $(\frac{|V|}{2} - 1)$-competitive algorithm exists for the online Q-ACTS problem.

III. Online Algorithm Design for Q-ACTS

To handle the challenges of the online Q-ACTS mentioned in Section I, we propose an online algorithm named **T**ransit **A**nchoring and **S**egment **C**oncatenation Algorithm (TASC) with a bi-criteria competitive ratio of $[1, O(\log |V|)]$. To this end, TASC introduces a novel notion, *REI*, denoted by $\mathcal{R}_r^k$, to evaluate the derived path of every candidate node $k \in T_r$ for each request $r \in R$, based on a *primal-dual analysis*. The larger $\mathcal{R}_r^k$ the node k leads to, the more resource-efficient its derived path is. Thus, as a request r arrives, TASC will reject it if $\max_{k \in T_r} \mathcal{R}_r^k < 0$ (i.e., there is no resource-efficient path) to tackle the first challenge. Once request r is admitted, the TASC will check the node k^* with the largest $\mathcal{R}_r^k$ to determine whether to use a transit node and which one should be used to address the second and third challenges. Specifically, if $k^* \notin \{s(r), t(r)\}$, the TASC will use k^* as a transit node; otherwise, the segment from $s(r)$ to $t(r)$ will be used directly.

A. Primal-Dual Analysis and Resource-Efficiency Index

We first subtly derive the relaxed LP of the Q-ACTS and its dual problem to assess the upper bound of the optimum. Specifically, we create new decision variable $x_r^k \geq 0, \forall r \in R, \forall k \in T_r$ and make 1) $d(r) \cdot x_r^k \leq z_r^k$ and 2) $\sum_{k \in T_r} p(r,k) \cdot x_r^k = y_r$. Due to constraint (5d), we know $y_r \leq 1$, leading to the new constraint (6b). In addition, we relax constraint (5c) to constraint (6c) by replacing z_r^k by $d(r) \cdot x_r^k$. Thus, we acquire the relaxed LP, which is an upper bound of (5a), as follows:

$$\max_{x_r^k} \quad \sum_{r \in R} d(r) \sum_{k \in T_r} p(r,k) \cdot x_r^k \tag{6a}$$
$$\text{s.t.} \quad \sum_{k \in T_r} p(r,k) \cdot x_r^k \leq 1, \quad \forall r \in R, \tag{6b}$$
$$\sum_{r \in R} \sum_{k \in T_r} h_r^k(u) \cdot d(r) \cdot x_r^k \leq m(u), \quad \forall u \in V, \tag{6c}$$
$$x_r^k \geq 0, \quad \forall k \in T_r,\ \forall r \in R. \tag{6d}$$

We then obtain the dual LP of the above relaxed LP, which further serves as the upper bound of (6a), as follows:

$$\min_{\alpha(r), \beta(u)} \quad \sum_{r \in R} \alpha(r) + \sum_{u \in V} m(u) \cdot \beta(u) \tag{7a}$$
$$\text{s.t.} \quad p(r,k) \cdot \alpha(r) + \sum_{u \in V} h_r^k(u) \cdot d(r) \cdot \beta(u) \geq d(r) \cdot p(r,k), \quad \forall r \in R,\ \forall k \in T_r, \tag{7b}$$
$$\alpha(r),\ \beta(u) \geq 0, \quad \forall r \in R,\ \forall u \in V. \tag{7c}$$

Note that $\alpha(r)$ and $\beta(u)$ are dual decision variables, and it can be envisaged that $\beta(u)$ indicates the overhead of node $u \in V$. By rearranging Eq. (7b), we have the following inequality:

$$\alpha(r) \geq d(r)\big(1 - \sum_{u \in V} \bar{h}_r^k(u) \cdot \beta(u)\big), \tag{8}$$

where $\bar{h}_r^k(u) = \frac{h_r^k(u)}{p(r,k)}$. Thus, to make the low-overhead high-probability (i.e., resource-efficient) routing path have a high REI, the REI is then defined as:

$$\mathcal{R}_r^k = 1 - \sum_{u \in V} \bar{h}_r^k(u) \cdot \beta(u). \tag{9}$$

Also, the TASC initially sets the dual variable $\beta(u)$ for each $u \in V$ to a small number, $\frac{1}{|V| \cdot (e-1)}$, implying that each node has no overhead. Moreover, to monitor the overhead of each node at any instant, the TASC updates the corresponding dual variable $\beta(u)$ of each node u on the routing path allocated for an admitted request r by the following formula.

$$\beta(u) \leftarrow \beta(u) \left[1 + \frac{\bar{h}_r^{k^*}(u) \cdot d(r)}{m(u)}\right]. \tag{10}$$

Later we show that the REI and the initialization and update rule of dual variables $\beta(u)$ are the cornerstones of the TASC to achieve the bicriteria competitive ratio.

B. Online Algorithm – TASC

The TASC includes four steps: 1) Dual Variable Initialization (DV Initialization), 2) Online Admission Control (OA Control), 3) Transit Node Selection (TN Selection), and 4) Dual Variable Update (DV Update). Specifically, DV Initialization initializes all dual variables in the beginning. Then, each time a request arrives, OA Control evaluates every trusted node by the REI and then admits it or not based on the REI. For each admitted request, TN Selection decides whether to use a transit node and which trusted node should be selected according to the REI. Finally, DV Update updates the corresponding dual variables if the request is admitted and waits for the subsequent requests.

Each step is described in an online manner as follows.

1) DV Initialization: DV Initialization initializes every dual variable $\beta(u)$ to $\frac{1}{|V|(e-1)}$ for each node $u \in V$. Note that the step is only conduced once in the beginning.

2) OA Control: As a request r arrives, OA Control calculates the REI $\mathcal{R}_r^k$ for each node $k \in T_r$ via Eq. (9). If $\mathcal{R}_r = \max_{k \in T_r} \mathcal{R}_r^k < 0$, OA Control will reject request r.

3) TN Selection: Following OA Control, once request r is admitted, TN Selection finds the node k^* with the maximum REI. That is, $k^* = \arg\max_{k \in T_r} \mathcal{R}_r^k$. If $k^* \in \{s(r), t(r)\}$, TN Selection directly uses $\ell(s(r), t(r))$ to serve request r with a demand size of $\lceil \frac{d(r)}{p(r,k^*)} \rceil$. Otherwise, it combines the two segments $\ell(s(r), k)$ and $\ell(k, t(r))$ to derive the routing path with a demand size of $\lceil \frac{d(r)}{p(r,k^*)} \rceil$ for request r.

4) DV Update: Following TN Selection, once the routing path for request r is determined, DV Update updates the corresponding dual variable $d(u)$ for each node u on the determined routing path according to Eq. (10).

The TASC is a $[1, O(\log |V|)]$-competitive algorithm. Due to the page limit, the pseudocode of the TASC and the proof are presented in Appendices D and E of [15], respectively.

Theorem 3. The TASC is a $[1, O(\log |V|)]$-competitive algorithm for the Q-ACTS.

C. Discussions for Practice Scenarios

1) Practical Heuristic: According to Theorem 3, the TASC might slightly deviate from the memory limit. To this end, we add one more step in the TASC to address the deviation. Specifically, the modified TASC will reject a request if any node on the path has insufficient memory to serve this request even if $\mathcal{R}_r^k \geq 0$. This modification performs well in practice.

2) Dynamic Traffic Flows: Section II-C assumes that each request transmit $d(r)$ data qubits periodically from $s(r)$ to $t(r)$ after arrival, but in practice, requests might departure. Thus, the system requires an update rule for terminating requests. Recall that as request r admitted by the system, $\beta(u)$ will be updated to $\beta(u)\left[1 + \frac{\bar{h}_r^{k^*}(u) \cdot d(r)}{m(u)}\right]$, Conversely, if r intends to leave the system, the central server can update $\beta(u)$ by:

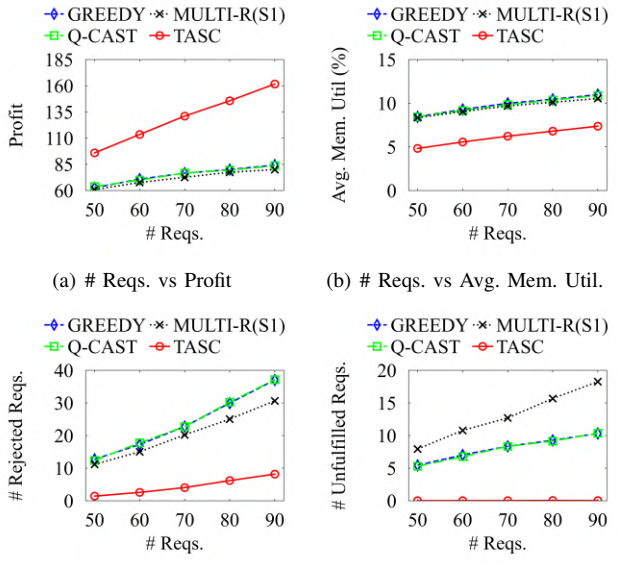
$$\beta(u) \leftarrow \frac{\beta(u)}{1 + \frac{\bar{h}_r^{k^*}(u) \cdot d(r)}{m(u)}}. \tag{11}$$

These adaptations enhance the TASC for practical applications.

IV. Performance Evaluation

A. Simulation Settings

Extensive simulations are conducted to compare the TASC with the GREEDY [8], Q-CAST [11] and MULTI-R(S1) [14]. Note that we made slight modifications to the GREEDY and Q-CAST to adapt them for online routing: for each arriving request, both the GREEDY and Q-CAST will be executed once to find a routing path. We vary one parameter for each simulation, with the default settings as follows. Following [12], we employ the Waxman model [17] to generate network topologies randomly with 100 quantum nodes across a 2000 km by 4000 km rectangular area. In this model, the probability that an edge exists between any two nodes u and v with the distance $l(u, v)$ is $\delta \cdot e^{-l(u,v)/\epsilon D}$, where δ and ϵ are tunable factors. We set $\delta = 0.9$, $\epsilon = 0.1$, and the average distance between two connected nodes is about 600 km. For trust relations, MEDICI [18], a social network generator, is used to construct the set of trusted nodes for each node, with a social density of 50%. The quantum memory limit of each node is randomly assigned from 62 to 72, which is a moderate number according to [19]. The qubit lifetime is set to $L = 7$ time slots. According to [8], the success entangling probability is $\mathrm{P}_{u,v} = e^{-a \cdot l(u,v)}$, where a is a material-dependent factor. With $a = 0.0002$, the success probability of entangling and swapping are both set to 0.8 [12]. By default, there are 50 online requests in total. Finally, we evaluate all methods by the following metrics: 1) *Profit*: the total demand size of fulfilled admitted requests. 2) *Average memory utilization*: the ratio of the resources used to serve requests to the total resources in the network. 3) *# Rejected Requests*: the number of requests rejected by the method. 4) *# Unfulfilled Requests*: the number of requests accepted but not fulfilled. In the figures, we abbreviate average, requests, utilization, memory as Avg., Reqs., Util., and Mem. respectively. Each result is average over 100 trials.



(a) # Reqs. vs Profit (b) # Reqs. vs Avg. Mem. Util.

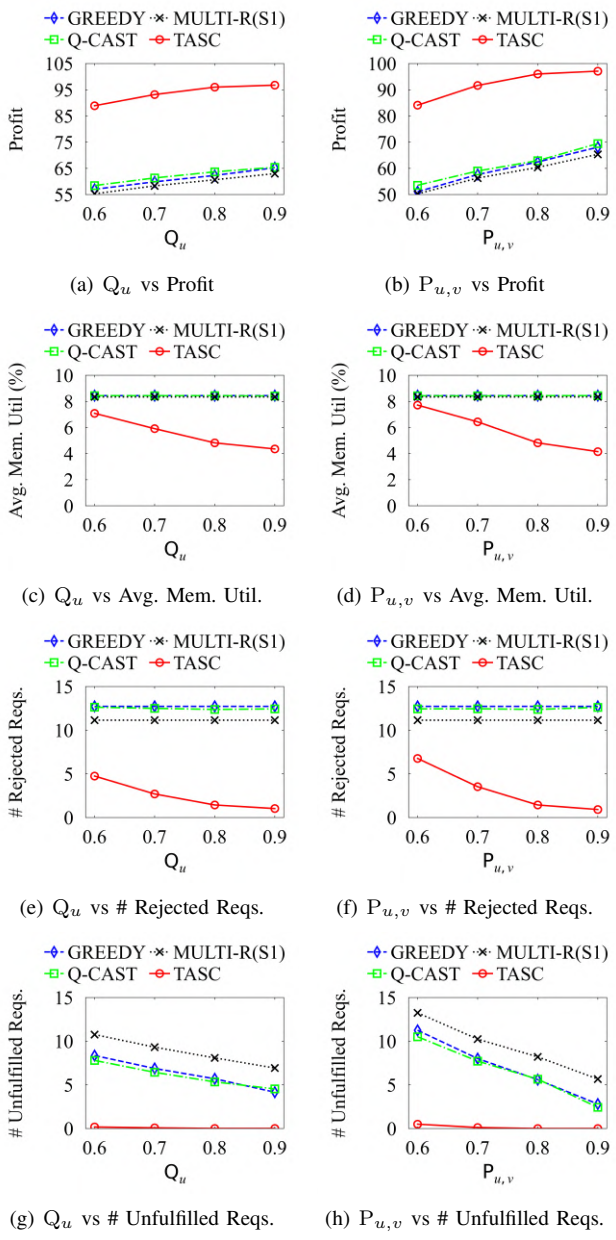
(c) # Reqs. vs # Reject Reqs. (d) # Reqs. vs # Unfulfilled Reqs.

Fig. 3. Effect on different metrics of the number of requests.

B. Numerical Results

Overall, the TASC outperforms all other approaches, as shown in Figs 3, and 4. TASC serves requests effectively and addresses all the challenges well. Due to the page limit, more experiment details can be found in Appendix F of [15].

1) Effect of Number of Requests: Fig. 3 illustrates the effect on different metrics of the number of requests. The TASC outperforms all other approaches across all metrics, particularly in effectively fulfilling most requests while maintaining a lower average memory utilization. The TASC performs better because it utilizes the REI to evaluate and derive the routing path based on the LP and primal-dual analysis. In contrast, the other methods may allocate massive resources to current requests, resulting in network congestion and competition for resources among requests in the future. Thus, it manifests that the TASC overcomes the first and third challenges and outperforms



(a) Q_u vs Profit (b) $P_{u,v}$ vs Profit

(c) Q_u vs Avg. Mem. Util. (d) $P_{u,v}$ vs Avg. Mem. Util.

(e) Q_u vs # Rejected Reqs. (f) $P_{u,v}$ vs # Rejected Reqs.

(g) Q_u vs # Unfulfilled Reqs. (h) $P_{u,v}$ vs # Unfulfilled Reqs.

Fig. 4. Effects on different metrics of entangling and swapping probabilities.

GREEDY, Q-CAST, and MULTI-R(S1) by up to 92%, 93%, and 98%, respectively, in profit.

2) Effect of Success Probability: Fig. 4 shows the effect of success probability of entangling and swapping. Specifically, Figs. 4(a)−4(d) demonstrate the effect of probability on profit and average memory utilization. As the two success probabilities increase, the number of established entangled paths also increases, leading to higher profits for each approach. Evidently, the TASC surpasses other methods due to its utilization of Theorem 1 in determining routing paths for each request. By examining the success probability of entangling and swapping, the TASC efficiently utilizes quantum memory, mitigating wastage and resolving the second challenge. Moreover, Figs. 4(e)−4(h) further analyze the rejected and unsolved requests for each approach. Figs. 4(e) and 4(f) demonstrate that the TASC results in significantly fewer rejected requests compared to other methods. This is because the TASC allocates quantum memory rationally, preventing all resources from being exhausted for serving current requests, thereby reducing future request rejections. Additionally, nearly all requests admitted by the TASC are successfully served. This once again highlights the REI's effectiveness for the TASC in achieving future-proof online admission control and addressing the first challenge.

V. Conclusions

In the paper, we study the limitations of existing architectures in QNs, particularly focusing on the issues of online decision-making for entangled paths of each request. To address these issues while ensuring successful transmission and data security, a new architecture OTER with an online optimization problem Q-ACTS is introduced. The Q-ACTS aims to maximize the number of served requests by anchoring appropriate transit nodes to detour hot spots. To this end, this paper proposes a novel $[1, O(\log |V|)]$-competitive online algorithm, named TASC. Finally, simulation results show that the TASC can outperform other approaches up to 98%.

References

[1] A. Khan *et al.*, "Software architecture for quantum computing systems—a systematic review," *J. Syst. Software*, vol. 201, p. 111682, 2023.

[2] S. Wang *et al.*, "Twin-field quantum key distribution over 830-km fibre," *Nat. photonics*, vol. 16, pp. 154–161, 2022.

[3] C. Portmann and R. Renner, "Security in quantum cryptography," *Rev. Mod. Phys.*, vol. 94, p. 025008, 2022.

[4] T. Metger *et al.*, "Security of quantum key distribution from generalised entropy accumulation," *Nat. Commun.*, vol. 14, p. 5272, 2023.

[5] X.-M. Hu *et al.*, "Progress in quantum teleportation," *Nat. Rev. Phys.*, vol. 5, pp. 339–353, 2023.

[6] J. Singh and K. S. Bhangu, "Contemporary quantum computing use cases: taxonomy, review and challenges," *Arch. Comput. Methods Eng.*, vol. 30, pp. 615–638, 2023.

[7] S. Pirandola *et al.*, "Fundamental limits of repeaterless quantum communications," *Nat. Commun.*, vol. 8, pp. 1–15, 2017.

[8] M. Pant *et al.*, "Routing entanglement in the quantum internet," *npj Quantum Inf.*, vol. 5, 2019.

[9] A. Farahbakhsh and C. Feng, "Opportunistic routing in quantum networks," in *IEEE INFOCOM*, 2022.

[10] K. Azuma *et al.*, "Quantum repeaters: From quantum networks to the quantum internet," *Rev. Mod. Phys.*, vol. 95, p. 045006, 2023.

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

[12] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *IEEE INFOCOM*, 2021.

[13] G. Zhao *et al.*, "Segmented entanglement establishment with all-optical switching in quantum networks," *IEEE ACM Trans. Netw.*, vol. 32, pp. 268–282, 2024.

[14] Y. Zeng *et al.*, "Entanglement routing design over quantum networks," *IEEE ACM Trans. Netw.*, vol. 32, pp. 352–367, 2024.

[15] W.-T. Ho *et al.*, "Online transit entanglement routing in quantum networks: Architecture design and optimization," Apr 2024. [Online]. Available: https://411410044tudgf.pse.is/5spshz

[16] N. Sangouard *et al.*, "Quantum repeaters based on atomic ensembles and linear optics," *Rev. Mod. Phys.*, vol. 83, pp. 33–80, 2011.

[17] B. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, pp. 1617–1622, 1988.

[18] D. F. Nettleton, S. Nettleton, and M. C. i. Farriol, "MEDICI: A simple to use synthetic social network data generator," in *MDAI*, 2021.

[19] O. D. J. Chow and J. Gambetta, "IBM quantum breaks the 100-qubit processor barrier," 2021. [Online]. Available: https://www.ibm.com/quantum/blog/127-qubit-quantum-processor-eagle

[20] H. Choi *et al.*, "Scalable quantum networks: Congestion-free hierarchical entanglement routing with error correction," *arXiv preprint arXiv:2306.09216*, 2023.
[21] Z. Li *et al.*, "Swapping-based entanglement routing design for congestion mitigation in quantum networks," *IEEE Trans. Netw. Service Manag*, vol. 20, pp. 3999–4012, 2023.
[22] S. Albers, *Competitive online algorithms*. Dept. Comput. Sci., Basic Res. Comput. Sci., Tech. Rep. LS-96-2, 1996.

APPENDIX A
RELATED WORK

QNs and their applications have attracted significant attention. For determining the routing path in QNs, Zhao *et al.* proposed a throughput maximization algorithm called REPS based on LP [12]. Zhao *et al.* introduced a SEE approach, where a segment, referred to nodes transmitted through the all-optical switching technique, can be connected using entanglement swapping [13]. Zeng *et al.* simultaneously maximized both the number of quantum-user pairs and their expected throughput [14]. Zhao *et al.* presented the EFiRAP approach, which prepares multiple candidate entanglement paths, determines purification schemes, and maximizes network throughput using an LP-based algorithm. However, all of these works utilize an LP solver to determine routing paths, which results in significant computational time and might exacerbate the limited lifetime issue of qubits. Pant *et al.* introduced a greedy-based algorithm that selects the entangled path with the minimum number of hops for each request [8]. Shi *et al.* presented a routing algorithm called Q-CAST, which is an modified Dijkstra algorithm, and designs recovery paths to address the failed entangled link establishment [11]. Choi *et al.* investigated the hierarchical aggregation of routing paths called QTN and examined the resource usage of QTN [20]. Li *et al.* introduced SPERF aiming at addressing congestion issues by reclaiming unused entanglement link resources and reallocating them for other requests [21]. However, despite efforts made by [21] to recycle idle network resources for congestion mitigation, these works still rely on shortest path-based methods to establish entanglement paths for requests, thereby failing to effectively address congestion issues at the algorithmic level. Overall, most of the above studies did not achieve online implementation and failed to effectively address congestion issues.

APPENDIX B
PROOF OF THEOREM 1

We first prove $F_1(\tau)$, which denotes the successful probabilities of path construction for request r directly using $\ell(s(r),t(r))$ at the time slot τ. Since the construction succeeds at the time slot τ, establishing $\ell(s(r),t(r))$ fails $(\tau-1)$ times. Thus,

$$F_1(\tau)=\Big(1-\Pr\big(\ell(s(r),t(r))\big)\Big)^{\tau-1}\Pr\big(\ell(s(r),t(r))\big).$$

On the other hand, for $F_2(\tau)$, representing the successful probabilities of path construction for request r via transit node k at the time slot τ. Similar to the proof of $F_1(\tau)$, we can assume that at the m-th time slot, segment $\ell(s(r),k)$ is successfully established. Since it takes at least one time slot to establish $\ell(k,t(r))$, m can be at most $\tau-1$. Thus,

$$\begin{aligned}
&F_2(\tau)\\
&=\sum_{m=1}^{\tau-1}\Big(1-\Pr\big(\ell(s(r),k)\big)\Big)^{m-1}\cdot\Pr\big(\ell(s(r),k)\big)\\
&\quad\cdot\Big(1-\Pr\big(\ell(k,t(r))\big)\Big)^{\tau-m-1}\cdot\Pr\big(\ell(k,t(r))\big)\\
&=\frac{1-\Pr\big(\ell(s(r),k)\big)\cdot\Pr\big(\ell(k,t(r))\big)}{1-\Pr\big(\ell(s(r),k)\big)}\Big(1-\Pr\big(\ell(k,t(r))\big)\Big)^{\tau-1}\\
&\quad\cdot\sum_{m=1}^{\tau-1}\Big(\frac{1-\Pr\big(\ell(s(r),k)\big)}{1-\Pr\big(\ell(k,t(r))\big)}\Big)^{m}. \qquad (12)
\end{aligned}$$

Finally, we can further extend Eq. (12) to its closed form.

$$\begin{aligned}
F_2(\tau)=&\frac{\Pr\big(\ell(s(r),k)\big)\Pr\big(\ell(k,t(r))\big)}{1-\Pr\big(\ell(s(r),k)\big)}\\
&\cdot\Big(1-\Pr\big(\ell(k,t(r))\big)\Big)^{\tau-1}\cdot\frac{P(1-P^{\tau-1})}{1-P}, \qquad (13)
\end{aligned}$$

where $P=\frac{1-\Pr(\ell(s(r),k))}{1-\Pr(\ell(k,t(r)))}$. If $P=1$, then

$$F_2(\tau)=\Pr\big(\ell(s(r),k)\big)^2\Big(1-\Pr\big(\ell(s(r),k)\big)\Big)^{\tau-2}(\tau-1). \qquad (14)$$

APPENDIX C
PROOF OF THEOREM 2

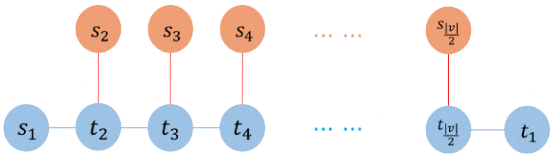


Fig. 5. Adversary network G.

We prove the theorem using an analysis technique with an adaptive online adversary [22]. The concept is that after the online algorithm makes a decision for the current request, the adversary can observe the algorithm's responses to all previous requests and generate the next request accordingly to maximize the performance gap between the offline optimum and the online solution. The adversary first constructs a graph $G=(V,E)$ and then generate several requests, each of which has a demand size d. In G, each quantum node $u\in V$ has limited memory $m(u)=3d-1$. At the first round, online adversary generates a request r_1 from s_1 to t_1 with only one path for selection, as shown in blue lines of Fig. 5. Thus, there exist only two possible outcomes: serving request r_1 or rejecting it. In response to these two outcomes, the online adversary will react differently. First, if the online algorithm serves r_1, the online adversary will generate each requests r_i from s_i to t_i at the $(i+1)$-th round, where $2\le i\le\frac{|V|}{2}$, as depicted in red lines of Fig. 5. Since the remaining memory of each node on the blue line (i.e., $3d-1-2d=d-1$) cannot support the requests in the subsequent time slots, these requests will be rejected. Thereby,

the offline optimum can serve requests $(r_2, r_3, \cdots, r_{\frac{|V|}{2}})$ at the later rounds, leading to a performance gap of $\frac{|V|}{2}-1$. Second, if the online algorithm rejects r_1 in the first time slot, the online adversary will not generate more requests, causing the gap to approach infinity. Overall, no matter the choices made by the online algorithm, the online adversary can respond accordingly, making it impossible for an $(\frac{|V|}{2}-1)$-competitive algorithm to exist for the online Q-ACTS. Therefore, the theorem follows.

APPENDIX D
PSEUDOCODE OF TASC

Algorithm 1 Pseudocode of TASC

Initialize $\beta(u) \leftarrow \frac{1}{|V|(e-1)}$, $\forall u \in V$;

As a request r arrives:

Calculate $\mathcal{R}_r^k \leftarrow 1 - \sum_{u\in V} \bar{h}_r^k(u) \cdot \beta(u)$ for each $k \in T_r$;
Set $\mathcal{R}_r^k \leftarrow \max_{k\in T_r} \mathcal{R}_r^k$;
if $\mathcal{R}_r^k < 0$ **then**
  Reject request r;
else
  Set $k^* \leftarrow \arg\max_{k\in T_r} \mathcal{R}_r^k$;
  if $k^* \in \{s(r), t(r)\}$ **then**
    Admit request r with a demand size of $\lceil \frac{d(r)}{p(r,k^*)} \rceil$ routed on $\ell(s(r), t(r))$;
  else
    Admit r with a demand size of $\lceil \frac{d(r)}{p(r,k^*)} \rceil$ routed on the path that consists of the $\ell(s(r), k)$ and $\ell(k, t(r))$;
    Update $\beta(u) \leftarrow \beta(u)\left[1 + \frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}\right]$ for each u on the routing path;

APPENDIX E
PROOF OF THEOREM 3

To prove the theorem, we have to 1) prove that the objective value of the solution output by the TASC (i.e., the primal solution) is at least as good as that of the optimum solution and 2) bound the deviation of the primal solution from the constraints.

For the first goal, we show how to construct a feasible dual solution $\mathcal{D}$ for the dual LP (i.e., (7a)) such that its value is no greater than that of the output by the TASC plus the total initial value of dual variables $\beta(u)$. To this end, in the constructed dual solution $\mathcal{D}$, we set $\alpha(r)$ to $d(r) \cdot \mathcal{R}_r^k$ if the TASC admits each request r at its arrival time (i.e., $\mathcal{R}_r^k \geq 0$); otherwise, $\alpha(r)$ is set to zero. Thus, we ensure that $\alpha(r) \geq 0$. In addition, each dual variable $\beta(u)$ in $\mathcal{D}$ is initialized to $\frac{1}{|V|(e-1)} > 0$ and will not decrease, guaranteeing that $\beta(u) \geq 0$. Therefore, the constructed dual solution $\mathcal{D}$ can satisfy constraint (7c). Subsequently, we examine constraint (7b) at the arrival time of request r in $\mathcal{D}$. If request r is admitted, then for any $k \in T_r$,

$$
\begin{aligned}
&p(r,k)\cdot\alpha(r) + \sum_{u\in V} h_r^k(u)\cdot d(r)\cdot\beta(u)\\
&= p(r,k)\cdot d(r)\cdot\mathcal{R}_r^k + d(r)\sum_{u\in V} h_r^k(u)\cdot\beta(u)\\
&= p(r,k)\cdot d(r)\cdot\max_{k\in T_r}\mathcal{R}_r^k + d(r)\sum_{u\in V} h_r^k(u)\cdot\beta(u)\\
&\geq p(r,k)d(r)\Big(1 - \sum_{u\in V}\frac{h_r^k(u)\cdot\beta(u)}{p_r^k}\Big) + d(r)\sum_{u\in V} h_r^k(u)\beta(u)\\
&= p(r,k)d(r) - d(r)\sum_{u\in V} h_r^k(u)\beta(u) + d(r)\sum_{u\in V} h_r^k(u)\beta(u)\\
&= p(r,k)\cdot d(r). \qquad (15)
\end{aligned}
$$

Eq. (15) shows that the constructed dual solution $\mathcal{D}$ can also satisfy constraint (7b), implying the *dual feasibility*. Afterward, we compare the solution of the TASC and the constructed dual solution $\mathcal{D}$. In the beginning, the constructed dual solution $\mathcal{D}$ has the total initial value of $\beta(u)$, i.e.,

$$\sum_{u\in V}\beta(u) = |V|\cdot\frac{1}{|V|(e-1)} = \frac{1}{e-1} < 1. \qquad (16)$$

Then, let $\Delta_r(u)$ denote the increase of dual variable $\beta(u)$ at the arrival time of request r. For each admitted request r at its arrival time, the dual solution $\mathcal{D}$ will increase by:

$$
\begin{aligned}
&\alpha(r) + \sum_{u\in V} m(u)\cdot\Delta_r(u)\\
&= \alpha(r) + \sum_{u\in V} m(u)\left[\beta(u)\Big(1 + \frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}\Big) - \beta(u)\right]\\
&= \alpha(r) + \sum_{u\in V}\bar{h}_r^{k^*}(u)\cdot d(r)\cdot\beta(u)\\
&= d(r)\left[1 - \sum_{u\in V}\bar{h}_r^{k^*}(u)\cdot\beta(u)\right] + \sum_{u\in V}\bar{h}_r^{k^*}(u)\cdot d(r)\cdot\beta(u)\\
&= d(r). \qquad (17)
\end{aligned}
$$

According to Eq. (17), we can know that when a request r is admitted, the ratio of the increase in the solution of the TASC to the dual solution $\mathcal{D}$ ultimately is:

$$\frac{d(r)}{\alpha(r) + \sum_{u\in V} m(u)\cdot\Delta_r(u)} = 1. \qquad (18)$$

Moreover, let ALG and OPT denote the solution output by TASC and the optimum solution of Q-ACTS. Then, by Eqs. (16) and (18), we know that

$$\mathcal{D} = ALG + \frac{1}{e-1} \Rightarrow ALG > \mathcal{D} - 1 \geq OPT - 1. \qquad (19)$$

Eq. (19) implies that $ALG \geq OPT$ since both ALG and OPT are integers, achieving the first goal.

Afterward, we proceed to achieve the second goal by showing that the solution of the TASC is nearly feasible for the primal problem. Let $M(u,r)$ denote the actually accumulated memory usage on node u after admitting or rejecting request r. Thus, the memory utilization on node u can be written as:

$$\rho(u,r) = \frac{M(u,r)}{m(u)}. \qquad (20)$$

Then, let $\beta(u,r)$ denote the value of the dual variable $\beta(u)$ after admitting or rejecting request r. To prove the *primal near feasibility* of the solution of the TASC, we first claim that the following inequality holds.

$$\beta(u,r) \geq \frac{e^{0.5\cdot\rho(u,r)}}{|V|\,(e-1)}. \qquad (21)$$

In this way, we can know that after admitting request r, for each $u \in V$ with $\bar{h}_r^{k^*}(u) \geq 1$ (i.e., request r is routed through node u),

$$\begin{aligned} e^{0.5\cdot\rho(u,r)} &\leq |V|(e-1)\cdot\beta(u,r) \\ &= |V|(e-1)\cdot\beta(u,r-1)\Big(1+\frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}\Big) \\ &\leq |V|(e-1)\Big(1+\frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}\Big) \\ &\leq 2|V|(e-1). \end{aligned} \tag{22}$$

The above second inequality holds since $\beta(u, r-1) \leq 1$ if request r is admitted by the TASC. Note that once node u with $\bar{h}_r^{k^*} \geq 1$ leads to $\beta(u, r-1) > 1$, then r must be rejected. That is because in such cases,

$$\mathcal{R}_r^k = 1 - \sum_{u\in V} \bar{h}_r^{k^*}\cdot\beta(u) < 1-1 = 0. \tag{23}$$

Therefore, by Eq. (22), we can know that

$$\rho(u,r) \leq 2\ln\Big(2|V|(e-1)\Big). \tag{24}$$

Finally, it suffices to show Eq. (21). We prove it by induction. For the basic case, when $r = 0$,

$$\beta(u,0) = \frac{1}{|V|(e-1)} = \frac{e^0}{|V|(e-1)} = \frac{e^{0.5\cdot\rho(u,0)}}{|V|(e-1)},$$

and thus Eq. (21) holds. Assume that it holds after admitting request $(r-1)$ by induction hypothesis. Then, for request r,

$$\begin{aligned} \beta(u,r) &= \beta(u,r-1)\Big(1+\frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}\Big) \\ &\geq \frac{e^{0.5\cdot\rho(u,r-1)}}{|V|(e-1)}\cdot\Big(1+\frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}\Big) \\ &= \frac{e^{0.5\cdot\rho(u,r-1)}}{|V|(e-1)}\cdot e^{\frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}} \\ &\geq \frac{1}{|V|(e-1)}\cdot e^{0.5\cdot\rho(u,r)}. \end{aligned} \tag{25}$$

The first inequality in Eq. (25) holds based on the induction hypothesis. Then, the second equality in Eq. (25) is correct since by the Taylor expansion, we can derive that

$$1+\frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)} \approx e^{\frac{\bar{h}_r^{k^*}(u)\cdot d(r)}{m(u)}},$$

since $m(u)$ is sufficiently larger than $\bar{h}_r^{k^*}(u)\cdot d(r)$.[2] Besides, the last inequality in Eq. (25) holds since request r utilizes $h_r^{k^*}(u)\cdot\lceil\frac{d(r)}{p(r,k^*)}\rceil$ units of memory on node u and

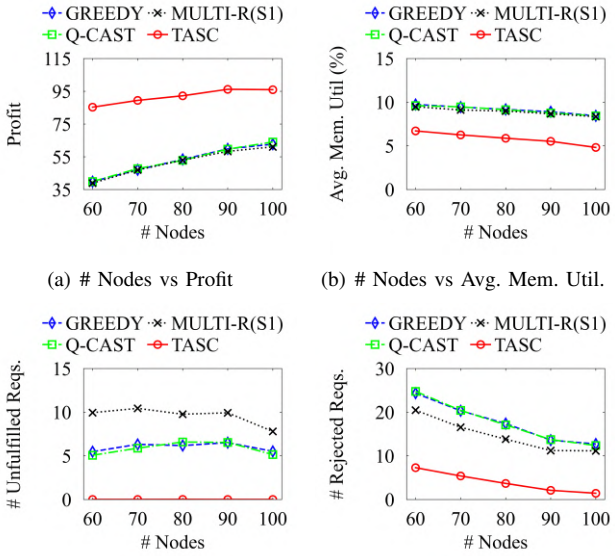
$$\begin{aligned} &0.5\cdot\rho(u,r-1) + h_r^{k^*}(u)\cdot\frac{d(r)}{p(r,k^*)}\cdot\frac{1}{m(u)} \\ &\geq 0.5\cdot\rho(u,r-1) + 0.5\cdot h_r^{k^*}(u)\cdot\lceil\frac{d(r)}{p(r,k^*)}\rceil\cdot\frac{1}{m(u)} \\ &= 0.5\cdot\rho(u,r), \end{aligned}$$

where $d(r) \in \mathbb{Z}^+$ and $0 < p(r,k^*) \leq 1$. Therefore, we achieve the second goal.

After achieving the two goals, we know that the TASC is a $\Big[1, 2\ln\Big(2|V|(e-1)\Big)\Big]$-competitive algorithm. The competitive ratio can be further expressed as $[1, O(\log|V|)]$. The theorem follows.

[2]It is reasonable to assume that $m(u)$ is sufficiently larger than $\bar{h}_r^{k^*}(u)\cdot d(r)$ since a request can typically be served and satisfied by a path in networks. Moreover, if not, such a request can be divided into multiple requests with a smaller demand in practice.

APPENDIX F
ADDITIONAL EXPERIMENTS



(a) # Nodes vs Profit (b) # Nodes vs Avg. Mem. Util.

(c) # Nodes vs # Unfulfilled Reqs. (d) # Nodes vs # Rejected Reqs.

Fig. 6. Effects on different metrics of the number of nodes.

We provide additional numerical results regarding the impact of the number of nodes in the network, as shown in Fig. 6.

1) Effect of number of nodes: The network includes more nodes within the same area, resulting in higher density. Consequently, more repeaters can be selected to serve requests, leading to increased network resources and success probability of entangling. Thus, as depicted in Figs. 6(a) and 6(b), with the increase in the number of nodes, the profit also increases, while the average memory utilization decreases. In these two metrics, compared to other methods, the TASC achieves better profits with significantly less memory utilization. Moreover, Figs. 6(c) and 6(d) further demonstrate the number of unsolved requests and rejected requests among all approaches. Specifically, Fig. 6(c) shows that once the TASC accepts a request, it completes the request regardless of the circumstances. However, for other methods, due to the lack of a robust admission control mechanism, the number of unsolved accepted requests is irregular. Fig. 6(d) illustrates that as the number of nodes increases, all approaches tend to reject fewer requests. Overall, the TASC significantly outperforms other baselines in various aspects due to its clever algorithm design.