Maximizing Entanglement Rates via Efficient Memory Management in Flexible Quantum Switches

Panagiotis Promponas, Víctor Valls, Saikat Guha, Leandros Tassiulas

***Abstract*—We study the problem of operating a quantum switch with memory constraints. In particular, the switch has to allocate quantum memories to clients to generate link-level entanglements (LLEs), and then use these to serve end-to-end entanglements requests. The paper's main contributions are (i) to characterize the switch's capacity region and study how it scales with respect to the number of quantum memories and probability of successful LLEs and (ii) to propose a memory allocation policy that is throughput optimal. In addition, when the requests are bipartite and the LLE attempts are always successful, we show that the proposed policy has polynomial time complexity. We evaluate the proposed policy numerically and illustrate its performance depending on the requests arrivals characteristics and the time available to obtain a memory allocation.**

I. Introduction

Quantum computing will transform the world by solving problems that are too complex for classical computers [2] (e.g., Shor's algorithm [3]). However, we are still nowhere near that day. Quantum programs of meaningful size require quantum computers with thousands of qubits [4], which is far from what quantum computers currently have [5], [6].

One way to increase the number of qubits of a quantum computer is to connect multiple quantum processors [7], [8], [9] with a quantum switch. A quantum switch is analogous to a classic packet switch, but its task is to create end-to-end entanglements with the clients it is connected. Figure 1 shows an example of how such a switch operates. The switch first allocates the limited quantum memories to clients/processors to generate link-level entanglements (LLEs)[1] with them (Figures 1a & b), and then it uses those to create end-to-end entanglements (Figures 1c & d).[2] The end-to-end entanglements are used by the quantum applications to, for example, teleport qubits or carry out distributed quantum operations (via non-local CNOT gates [10]).

Quantum networking is in its infancy since single-hop communications are still challenging [13]. However, the building blocks of how quantum networks will operate already exist, prompting researchers to start designing the algorithms that will run the networks when the hardware becomes available [14], [15], [16], [17]. Regarding quantum switches, previous work has studied their operation under a variety of settings [18], [19], [20], [21], [22]. In brief, [18] and [19] study an idealized switch with bipartite and tripartite end-to-end entanglements requests when the request arrivals are symmetric and decoherence [11] is negligible. The work in [20] studies a quantum switch with bipartite requests when there is no memory decoherence and LLE attempts succeed probabilistically. The contributions of [20] are to characterize the switch capacity region and to propose on-demand policies that are throughput optimal. Similarly, the recent work in [21] extends the setting in [20] to capture that LLEs expire (i.e., "decohere") after some time in practical systems. In [22] the authors consider a quantum switch constrained by quantum memory limitations, assuming that while clients do not explicitly request end-to-end entanglements, they constantly require such. Moreover, in contrast to our work, they assume that the quantum memories are dedicated to clients. This setting differs from our work, where we propose that a quantum switch with limited memory can enhance its capacity by dynamically reallocating memory in response to network needs. To the best of our knowledge, this work is the first to study the capacity of a quantum switch that operates with limited memory and is capable of redistributing the memories to different channels/clients at each time interval.

In summary, in this paper we study the problem of operating a quantum switch when it can store a limited number of qubits. In particular, the switch has a limited number of quantum memories and it can decide how to allocate quantum memories to generate LLEs. Studying this problem is important because memory is a scarce resource in practical quantum systems. To this end, this paper makes the following contributions:

- We present the first physical and mathematical model of a quantum switch that has to operate with a limited amount of quantum memories which are capable of being redistributed at the start of each time interval (Section II). Our model allows LLEs to decohere and end-to-end entanglement requests to be multipartite.
- We characterize the capacity region of the quantum

The research work was supported by the Army Research Office under the project number W911NF2110325 and by the National Science Foundation under project numbers EEC-1941583 CQN ERC and CNS 1955744. A partial and preliminary version of this paper appeared in 2023 IFIP Networking Conference (IFIP Networking) [1].

P. Promponas and L. Tassiulas are with the Department of Electrical Engineering, Yale University, New Haven, CT, USA (email: {panagiotis.promponas, leandros.tassiulas}@yale.edu).

V. Valls is with IBM Research Europe – Dublin (email: victor.valls@ibm.com).

S. Guha is with the Wyant College of Optical Sciences, The University of Arizona, Tucson, AZ, USA (email: saikat@arizona.edu).

[1]Also known as EPR pairs. A LLE or EPR pair consists of two entangled qubits [10]. One qubit at the switch and the other qubits at the client.

[2]An end-to-end entanglement is created by performing a measurement (BSM or GHZ) on the qubits at the switch [11]. The process is also known as entanglement swapping when the requests are bipartite [12], [13].

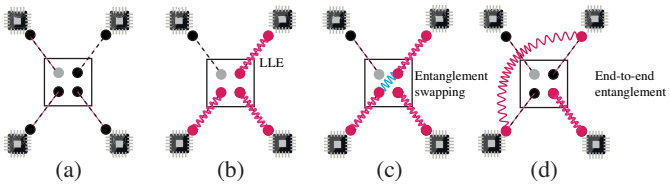


Figure 1. Illustrating the operation of a quantum switch with three quantum memories and four clients, each equipped with a single quantum memory. (a) The quantum memory registers should be allocated to the clients for the LLE attempts (b) LLEs are created between the switch and the clients with assigned memories. (c) The switch performs an entanglement swapping operation. (d) An end-to-end entanglement is created as a result. An entanglement swapping operation consists of performing a joint BSM measurement with the qubits at the switch.

switch with memory constraints (Section III-A), i.e., the set of arrival rate of requests for end-to-end entanglements for which there exists a scheduling policy that stabilizes the switch.

- We study how the capacity region depends on the number of quantum memories and the probability of successful LLEs (e.g. by increasing the quality of the quantum hardware used). This analysis is necessary when designing a quantum switch (e.g. we know the traffic the switch needs to satisfy and we have to decide upon how many quantum memory registers the switch has to possess).
- We propose a memory allocation policy (MEW) that stabilizes the switch when (i) the LLEs last one time slot and (ii) the arrivals of requests are in the interior of the capacity region (Section IV-A). Finding a throughput optimal policy in this setting is challenging because the admissible scheduling decisions depend on the memory allocation. Therefore, the connectivity of the switch in every time slot is not determined by an i.i.d random variable as it is the case in classical networking problems [23] (see discussion after Theorem 1).
- We present MEW2, a variant of MEW tailored to the case where end-to-end entanglements are bipartite and LLE attempts are always successful. This case is important since multipartite requests can be divided into multiple bipartite requests (universality of two-qubit gates [11]) and because, by appropriately extending the duration of a time slot to perform entanglement distillation and retrials, we can generate LLEs almost surely [24]. MEW2 finds a memory allocation by finding a special matching in the N-complete graph. Finding such matching has polynomial time complexity. When the assumptions under which MEW2 is optimal are violated, this policy suggests a heuristic variant of MEW (see Section VII-D).

Finally, in Section VII, we evaluate the proposed policy numerically and illustrate its performance depending on the requests arrivals and the time available to obtain a memory allocation.

II. Quantum Switch Model & Operation

In this section, we demonstrate how a quantum switch operates in the physical layer and we abstract this model mathematically enabling its study and operation. The first subsection (Section II-A) describes the physical model of the quantum switch and motivates our mathematical framework. In fact, we abstract this physical model in the next subsections (Sections II-B and II-C) to make it more suitable for the application of control and optimization techniques.

A. Quantum switch physical model

The actual hardware that is going to be employed for large scale quantum networks is yet unknown. However, the solid-state quantum emitters have recently emerged as promising candidates [25]. They have a number of advantageous properties, including electronic spin qubits with long coherence times [26], [25], fast gates, access to nuclear qubit registers, deterministic qubit fabrication, and operating temperatures [25]. The most daunting obstacle in scaling quantum information processing is generating high-fidelity entanglement between spatially separated defects. Entanglement mediated by photons stands out as a unique mechanism for long distance entanglement even across room-temperature environments [27]. In the following we describe how a quantum switch can be implemented by using solid-state quantum emitters [25].

A qubit is an abstract concept that has different ways in which one can store and process it. Three examples that offer different properties are nuclear-spin qubits, photon qubits and electron spin qubits (atomic). A quantum memory consists of an electron spin qubit centered around neighboring nuclear-spin qubits. We can excite the electron spin qubit to generate a photonic qubit entangled with it. A nuclear-spin qubit can hold a qubit state for more time than an electron spin qubit. Figure 2 illustrates the physical prossess and implementation of a quantum switch using solid-state quantum emmiters and photonic qubits for the distribution of the entanglement.

We consider that the time is slotted, and each time slot has a duration of $\mathcal{T}$ seconds. In the rest of this section we describe the operation of the quantum switch, in every time slot on the physical level, describing Figure 2. We denote Δ to be a tuning parameter that affects the duration of a time slot, i.e, $\mathcal{T}$. Specifically, Δ denotes the time that the switch starts the BSMs in order to satisfy clients' requests for end-to-end entanglement. The time from the start of the time slot until Δ, the switch tries to establish a successful LLE. Each LLE succeeds with a certain probability that depends on the protocols used to establish the entanglement between the switch and the client. We abstract our model to be independent of the actual procedures used to generate the LLE and thus we only care about the probability for the LLE being successful.

Let the fidelity of a successful LLE be F_0. After its generation, the LLE's fidelity degrades until its final consumption at time $\tau = \mathcal{T}$ whereas it also degrades because of the noise induced by the BSM/GHZ projection that occurs at time $\tau = \Delta$. Let the fidelity of the end-to-end entanglement after the BSM/GHZ projection be $F^{'} < F_0$. For now, we assume that the fidelity $F^{'}$ suffices for the application that requested the end-to-end entanglement. However, as will be discussed later in Section VI, when more than one memory is associated with a client and the client possesses more than one memory, a switch might decide to perform entanglement distillation to its

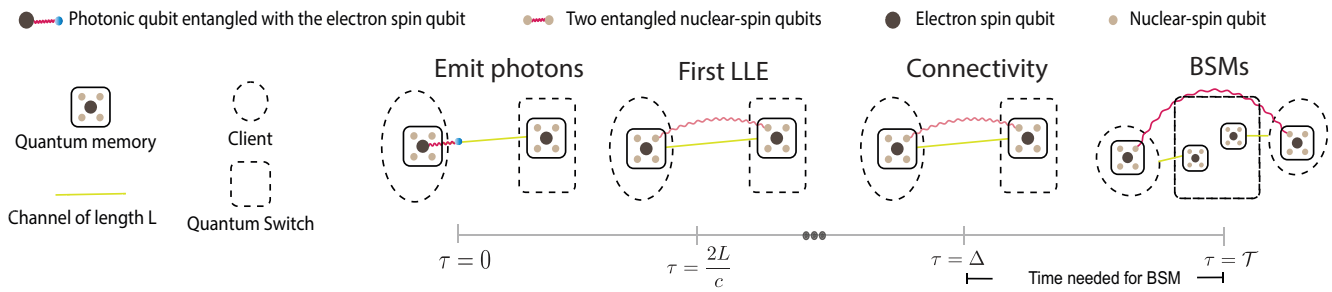


Figure 2. Operation of a quantum switch on the physical level in a time slot of duration $\mathcal{T}$. In the figure, L denotes the length of the channel between the client and the switch, c denotes the speed of light and Δ refers to the time that the switch determines which BSMs to execute based on the available LLEs.

LLEs. In that way, the LLEs that have not been destroyed or consumed by the distillation process and thus will be available at time $\tau = \Delta$, will have greater fidelity.

Before analyzing a time slot, we discuss some assumptions that are made for the rest of the paper.

Assumption 1. *There is one channel per client and we can assign one memory to each channel.*

This assumption goes without loss of generality since we can modify the model appropriately to include multiple channels. In fact, our model can be modified to optimally allocate memories to clients even if we can allocate more than one memory to each and more than one memory to a specific channel. Assumption 1 implies that we can allocate at most 1 quantum memory to each client. A detailed discussion on how Assumption 1 can be relaxed follows in Section VI.

Assumption 2. *Successful LLEs can be used during the time slot they are created.*

This assumption is relevant to the decoherence time of an entanglement and is frequently used in prior research (e.g., [21]). Essentially, we assume that after one time slot from their creation, the entangled qubits should be discarded since due to loss of fidelity they are useless to the end user applications. This assumption facilitates the synchronization of the duration for which a quantum entanglement is maintained with the operational intervals at which a quantum switch administers quantum memory allocations and addresses requests. While this paper presents a model based on the fact that entanglements do not survive beyond a single round, it also acknowledges the potential for a more generalized model where entanglements are available for several rounds. In such a framework, the policies developed herein could be adapted as heuristics. Within this heuristic application, our scheduling policies would still assume that unused LLEs would expire in the next time slot. However, the decision-making process of our memory allocation framework for the subsequent time slot could account for increased probability of generating a successful LLE should the corresponding memory stay in the same link with an alive qubit.

Below we analyze a time slot from its beginning until the end ($0 \leq \tau \leq \mathcal{T}$). Our model is abstract enough to be independent of the implemented physical protocols that perform the error correction and entanglement distillation. This level of abstraction is motivated by the classical networking counterpart where the network scheduling and routing decisions are unaware of the actual physical procedures, rather they depend on some parameters that indicate the quality of the link, the probability of successful packet transmission and more. Note that the following discussion regarding Figure 2 assumes control decisions that have already been made, i.e., assigned quantum memory to the client as shown and BSM operation to be executed in $\Delta \leq \tau \leq \mathcal{T}$. The optimization of such control variables is the main goal in the rest of the paper.

Start ($\tau = 0$): The first step is to excite an electron spin qubit to generate a photonic qubit entangled with it. Although there are multiple ways to achieve that, in this paper we do not need to assume a specific one. As the photonic qubit travels through the link, it goes through loss. This loss can be heralded and therefore translated into a probability that a LLE succeeds.

Around every electron spin qubit is a collection of nuclear-spin qubits that can store the qubit for longer than an electron spin qubit. The moment we initialize an electron spin qubit, we perform a swap gate between this and a neighboring nuclear-spin qubit in order to store the entanglement to the more coherence nuclear-spin qubit.

Shooting photonic qubits ($0 \leq \tau \leq L/c$): The photonic qubit will reach the switch after L/c seconds, i.e., the photon passes the channel of length L at the speed of light. Since we can excite the electron spin qubit more than once in that time, each client can periodically send photons that are initially entangled with the electron spin qubit and use blind distillation in the nuclear-spin qubit [28]. In that way, we can increase the fidelity of the LLE when such succeeds.

Arrival of first photon ($\tau = L/c$): At $\tau = L/c$, the first photonic qubit reaches the switch and gets stored in a nuclear-spin qubit in an assigned dedicated quantum memory. At that point, a classical bit is transmitted back to the client to confirm the success or failure of the entanglement.

Entanglement Distillation or retrial ($2L/c \leq \tau \leq \Delta$): Δ is a tuning parameter that defines at what time the switch performs the BSMs/GHZ projections to satisfy requests between clients. Note that Δ should be greater or equal than $2L/c$ even if the switch does not perform distillation, since the classical bit should reach back to the client. Therefore, from $2L/c \leq \tau \leq \Delta$, a client can either retry the generation of LLE in case it did not succeed, or perform entanglement distillation in case it succeeded. Although depending on the protocol we can retry when a LLE fails, we define the probability of successful LLE to be the probability of having a successful

entanglement (with sufficient fidelity) *at time* $\tau = \Delta$.

Bell state measurements (or GHZ projections) ($\Delta \leq \tau \leq \mathcal{T}$): In this time period, the switch performs the Bell state measurements to satisfy requests for end-to-end entaglements. At $\tau = \mathcal{T}$, nuclear qubits of served clients are entangled with each other. The entanglements that were not used until $\mathcal{T}$ decohere and are not available for the next time slot (Assumption 2).

In our model we assume limited quantum memory and that we can rearrange the memories inside the switch to increase its capacity. However, a question that might arise is whether by this rearrangement we waste entangled photonic qubits that the clients can send to the switch and therefore end up to inefficient results. However, to see that this is not the case we distinguish between two cases: a quantum memory in a client (a) was or (b) was not able to generate a LLE. In the first case, the client cannot use that nuclear-spin since it is occupied. In the latter case, the client can retry to generate a LLE not only until $\tau < \Delta$ but also during the slot $\Delta \leq \tau \leq \mathcal{T}$. Note that the latter time interval is not negligible. Essentially, such client can utilize the corresponding channel by sending entangled photonic qubits to achieve a LLE with the switch during the next time slot. Our model, is expressive enough to include these retrials by assuming an increased probability of successful LLE in the next time slot in case the memory stays associated with the same client/channel. As it will become apparent during the mathematical analysis in the rest of the paper, this increase in the probability in case the previous attempt was unsuccessful is permitted in our model since it does not depend on the control actions.

Our initial model's depiction of entanglement generation, which featured qubit transmission from one client to the switch, is one example within a broader range of entanglement generation schemes our mathematical framework can accommodate. In particular, emission based schemes where photons from two different nodes are brought to interference at a halfway point to create entanglement [27], can be seamlessly integrated into the abstract mathematical model that will be described in the paper. Our model does not require granular details of the photonic interactions but instead uses the probability of successful LLEs as a pivotal parameter. This approach allows us to focus on what is essential for the optimization layer—the likelihood of achieving functional entanglement within each time slot—irrespective of the physical implementation. This level of abstraction allows our model to be applicable to diverse scenarios in quantum communication without the need for reconfiguration to fit specific entanglement generation methods.

B. Switch model abstraction and operation overview

In this section, we abstract the previous physical model into a mathematical formulation, more suitable for describing optimization techniques and algorithms.

We consider a quantum switch with M quantum memory registers[3] and N clients that operates in slotted time. In each time slot $t = 1, 2, 3, \ldots$, the switch receives a vector of requests $A(t) = (A_1(t), \ldots, A_R(t))$, where $A_r(t) \in \{0, 1\}$ for all $r \in \{1, \ldots, R\}$. The type of request, r, explicitly determines the subset of clients involved in a given request $A_r(t)$. Specifically, a request $A_r(t)$ involves connecting two or more clients (i.e., it is multipartite), and we use set $\Omega(r) \subseteq \{1, \ldots, N\}$ to denote the clients that participate in a request. For example, $\Omega(r) = \{1, 2\}$ if a request of type r connects clients 1 and 2.

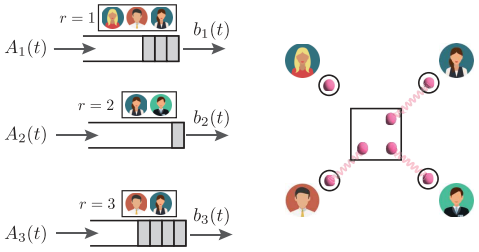


Figure 3. Example of a quantum switch with three types of requests. The switch is connected to three users, which only allows it to serve requests of type 2 and 3.

Upon arrival, the requests are stored in separate queues $Q(t) = (Q_1(t), \ldots, Q_R(t))$ to await service. The queues evolve as $Q(t+1) = [Q(t) - b(t)]^+ + A(t)$ where $[\cdot]^+ := \max\{0, \cdot\}$ and $b(t) = (b_1(t), \ldots, b_R(t))$ indicates the requests served in time slot t. In particular, $b_r(t) = 1$ if a request $r \in \{1, \ldots, R\}$ is served, and $b_r(t) = 0$ otherwise.

The switch's task is to serve as many requests as possible subject to operational constraints. In particular, the switch can only serve a request if all the clients that participate in it have an active LLE. Figure 3 shows an example of a switch with four clients and three types of requests $r \in \{1, 2, 3\}$. Observe that the switch can serve requests of type 2 and 3, but not of type 1 as one of the clients is not connected with the switch.

In the next section, we describe how the switch allocates quantum memories to clients and how that affects the switch connectivity and the set of admissible service vectors.[4]

C. Switch operation and decision variables

In each time slot, the quantum switch performs three types of actions. It (i) allocates quantum memories to clients; (ii) generates LLEs; and (iii) serves multipartite requests using the LLEs. A LLE can be used to serve *one* request as this is consumed to generate end-to-end entanglement [19]. Next, we describe the control variables that the switch selects in each time slot.

1) Quantum memory allocation: Since $M < N$ (Assumption 1), the switch has to decide how to allocate memories to clients. We use $m_n(t)$ to denote whether the switch assigns a quantum memory to a node $n \in \{1, \ldots, N\}$ in time slot t,

[3] As described in section II-A they consist of an electron spin qubit centered around multiple nuclear-spin qubits.

[4] i.e., the requests that can be served given a switch connectivity.

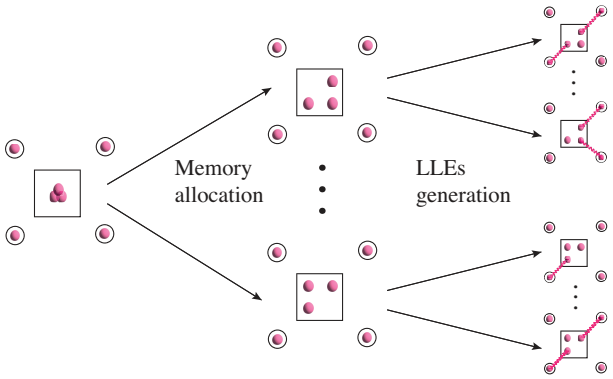


Figure 4. Illustrating how the quantum memory allocation results in different possible connectivities in a quantum switch with $M = 3$ and $N = 4$. Observe from the figure that different memory allocations can result in the same switch connectivity.

and collect these in vector $m(t) = (m_1(t), \ldots, m_N(t))$. The set of eligible memory allocations $\mathcal{M}$ is given by

$$\mathcal{M} = \Bigg\{ (m_1, \ldots, m_N) : m_n \in \{0,1\} \ \forall n \in \{1, \ldots, N\} \text{ with } \sum_{n=1}^{N} m_n \leq M \Bigg\}.$$

2) LLEs generation and switch connectivity: After the memory allocation, the switch has to generate LLEs with the clients that are connected to a memory. The switch attempts to create LLEs by sending entangled qubits (e.g., photons) over a fiber-optical channel, but only a fraction of the LLE attempts are successful due to interference (see section II-A). Also, LLEs last for a limited amount of time due to a phenomenon known as decoherence [29] (Assumption 2).

We model the switch connectivity in a time slot as follows. Let $p_n \in [0,1]$, $n \in \{1, \ldots, N\}$ be the probability that a LLE attempt succeeds. Vector $k(t) = (k_1(t), \ldots, k_N(t))$ with

$$k_n(t) = \begin{cases} 0, & m_n(t) = 0, \\ 0, & m_n(t) = 1 \quad \text{w.p.} \quad 1 - p_n, \\ 1, & m_n(t) = 1 \quad \text{w.p.} \quad p_n \end{cases} \tag{1}$$

denotes the collection of successful LLEs in a time slot, i.e., the switch's connectivity. We use set $\mathcal{K}(m(t)) \subseteq \{0,1\}^N$ to capture all the possible switch connectivities for a given memory allocation $m(t) \in \mathcal{M}$. Note that a memory allocation has a total of $|\mathcal{K}(m)| = 2^M$ possible switch connectivites if all the memories are used.[5] We assume that in every time slot all the memories are used since there is no cost associated with the allocation of a memory to a client. Figure 4 shows how the switch connectivity depends on different memory allocations and the successful LLEs. Also, observe from the figure that different memory allocations can result in the same connectivity due to some LLE attempts failing.

3) End-to-end entanglement requests service: The switch connectivity in a time slot affects the set of available service vectors. Let $k(t) \in \mathcal{K}(m(t))$ with $m(t) \in \mathcal{M}(t)$ be the switch connectivity at time slot t. The set of admissible service vectors is given by:[6]

$$\mathcal{B}(m(t), k(t)) = \Bigg\{ b_r \in \{0,1\}, r \in \{1, \ldots, R\} : \text{there exists a matrix } S \in \{0,1\}^{R \times N} \text{ s.t. } s_{rn} = 1 \text{ for all } n \in \Omega(r) \text{ iff } b_r = 1, \text{ and } \sum_{r=1}^{R} s_{rn} \leq k_n(t) \ \forall n \in \{1, \ldots, N\} \Bigg\}.$$

That is, $\mathcal{B}(m(t), k(t))$ contains a collection of binary vectors, where the r'th entry of a vector is equal to one if and only if (i) all the clients involved in a request of type r have an active LLE with the switch, and (ii) a LLE is used to serve one request.

III. Capacity Region and the Effect of Quantum Memory and Link Quality

In this section, we characterize the capacity region of the quantum switch (Section III-A), and study the effect of memory and link quality in the switch's capacity (Section III-B).

A. Capacity region

Before designing an algorithm, we need to characterize the set of arrival rates that the switch can support. To start, let $\lambda := \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} A(t)$ be the long-term arrival rate of requests at the quantum switch. We say an arrival vector λ is admissible (or, it can be supported) if there exists a policy π that can generate a sequence of service rate vectors $\{b^\pi(t)\}_{t=1}^{\infty}$ such that

$$\lambda_r \leq f_r^\pi := \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} b_r^\pi(t) \qquad \forall r \in \{1, \ldots, R\}. \tag{2}$$

That is, for a given vector λ, the switch must be able to generate a long-term service vector f^π that is equal to or larger than λ component-wise.

To define the switch capacity region, we decouple the decision variables from the time slot index t and express them as the fraction of time they can occur. In short, let θ_m denote the fraction of time a memory allocation $m \in \mathcal{M}$ is used, and $\mathbb{P}(k; m)$ the probability that a switch connectivity $k \in \mathcal{K}(m)$ occurs for a given a memory allocation $m \in \mathcal{M}$. Similarly, let $\delta_b^{k,m}$ be the fraction of time that each service vector $b \in \mathcal{B}(m, k)$ is used for a given switch connectivity and memory allocation. We have the following proposition.

[5] Otherwise, the switch has $2^{\sum_{r=1}^{R} m_r(t)}$ possible connectivities.

[6] Although the set $\mathcal{B}(m(t), k(t))$ depends only on the network connectivity, $k(t)$, we parameterize it with $m(t)$ as well to emphasize that the service vectors are picked after the memory allocation.

Proposition 1 (Quantum switch capacity region)**.** *The capacity region of the quantum switch is:*

$$\Lambda := \Bigg\{ f^{\pi} : f^{\pi} = \sum_{m\in\mathcal{M}} \theta_m \sum_{k\in\mathcal{K}(m)} \mathbb{P}(k;m) \sum_{b\in\mathcal{B}(m,k)} \delta_b^{m,k} b, \quad \sum_{m\in\mathcal{M}} \theta_m = 1, \ \sum_{b\in\mathcal{B}(m,k)} \delta_b^{m,k} = 1, \quad \theta_m \geq 0, \ \delta_b^{m,k} \geq 0, \quad \text{for all } b \in \mathcal{B}(m,k), \ k \in \mathcal{K}(m), \ m \in \mathcal{M} \Bigg\}. \tag{3}$$

Proof sketch: The full proof is omitted due to space constraints. However, it follows the same methodology as in [23], [30]: writing the fraction of time that the service vectors can be generated—depending on the memory allocations and switch connectivities in our case. ■

Note that if $\lambda \in \Lambda$ (i.e., the long-term average of requests arrivals is in the capacity region), then there exists a vector f^{π} that satisfies (2). Having $\lambda \in \Lambda$ is usually known as *the necessary condition* for having stable queues [23].

B. Effect of Memory and Link Quality in the Switch Capacity

This section investigates how the number of quantum memories in the switch and the probability of successful LLE for each channel influence the overall throughput. Such analysis facilitates the optimal design of future quantum hardware used for the implementation of quantum switches.

We distinguish the capacity region of a quantum switch with M available memories as $\Lambda^{(M)}$. Similarly, we define as $\mathcal{M}^{(M)}$ the possible memory allocations when we have M memories available. For simplicity, in this section we focus only on bipartite requests and when the probability for LLEs being successful in every link are equal to p.

Let $\mathcal{P}(k)$ and $\mathcal{Y}(k)$ be the set of matchings and perfect matchings of the complete graph that results from the clients $n \in \{1,\dots,N\}$ for which $k_n = 1$, i.e., the clients that successfully generated a LLE. Also, let $co(\cdot)$ define the convex hull of a set.

Under the bipartite requests assumption, the capacity boundary can be rewritten as:

$$\Lambda^{(M)} := \Bigg\{ f^{\pi} : f^{\pi} = \sum_{m\in\mathcal{M}^{(M)}} \theta_m \sum_{k\in\mathcal{K}(m)} p^{|k|}(1-p)^{M-|k|} x^{m,k}, \quad \sum_{m\in\mathcal{M}} \theta_m = 1, \ \theta_m \geq 0, \ x^{m,k} \in co(\mathcal{Y}(k)) \quad \text{for all } k \in \mathcal{K}(m), \ m \in \mathcal{M}^{(M)} \Bigg\}. \tag{4}$$

Where $|k|$ represents the number of LLEs that were successful out of the M at connectivity k. Note that instead of $co(\mathcal{P}(k))$ we use $co(\mathcal{Y}(k))$ for the domain of $x^{m,k}$ in $\Lambda^{(M)}$ to consider the points in the boundary of the capacity region where the rate vectors cannot be increased in any component and still be in the capacity region.

Note that in the calculation of the capacity in (4), we utilized the fact that $\mathcal{B}(m,k) = \mathcal{Y}(k)$. This principle is based on the premise of bipartite requests and Assumption 1. It's crucial to recognize that, although Assumption 1 is without loss of generality for the rest of the paper as discussed in Section VI, the specific conclusions drawn in this section are fundamentally reliant on the concept that potential service vectors correspond to matchings in a complete graph.

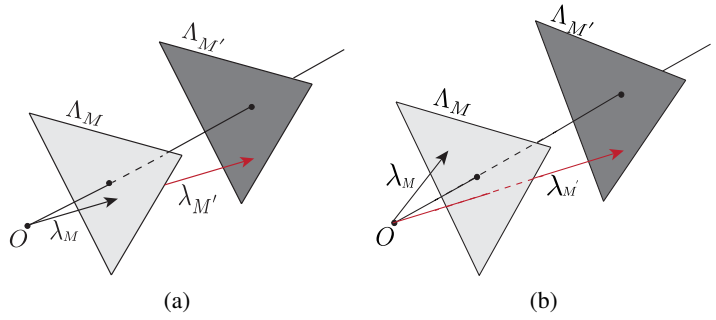


Figure 5. Illustrating the capacity regions $\Lambda^{(M')}$ and $\Lambda^{(M)}$ with M' and M number of quantum memories respectively. In the figure, the vectors λ_M and $\lambda_{M'}$ illustrate (a) two co-linear rate vectors i.e., $\lambda_{M'} = \alpha\lambda_M$ for $\alpha > 1$, and (b) two arbitrary rate vectors on the capacity regions. We are interested in how "longer" the rate vector on the "larger" capacity region is.

Since the capacity region of a quantum switch is a convex polytope in a possibly high dimensional space, it is challenging in general to compare $\Lambda^{(M)}$ and $\Lambda^{(M')}$ for $M \neq M'$. However, as we will show in Lemma 1, in our setting the comparison is possible since every vector in the boundary of the capacity region yields the same ℓ_1 norm. Hence, given a specific number of quantum memories and a probability for successful LLEs, the total throughput maintainable by the switch does not depend on the ratio of the requests' average arrivals. Therefore, one can optimize the total throughput without worrying what the actual arrival rate would be.

A vector λ_M in $\mathbb{R}^{N^2}$ corresponds to a target rate vector, i.e., the r'th component of λ_M corresponds to the target service rate of the request r. Thereby, it is intuitive to compare $\Lambda^{(M')}$ and $\Lambda^{(M)}$ according to vectors $\lambda_{M'}$ and λ_M that lie on them. In that way, we can quantify how much we can extend an arrival rate vector when we increase the number of quantum memories in the switch. For such a comparison, we use the ℓ_1 norm, which captures the total throughput.

Figure 5 illustrates the capacity regions $\Lambda^{(M)}$ and $\Lambda^{(M')}$ with M and $M' > M$ number of quantum memories respectively. In Figure 5a, we illustrate two co-linear rate vectors i.e., $\lambda_{M'} = \alpha\lambda_M$ for $\alpha > 1$, whereas in Figure 5b we show two arbitrary rate vectors on the capacity regions. In the former, we can quantify how much we can extend an arrival rate vector when we increase the number of quantum memories in the switch. We are interested in how "longer" the rate vector on the "larger" capacity region is.

For the statement of the main results of this section we present the following facts.

Fact 1. $\|x\|_1 = \lfloor \frac{|k|}{2} \rfloor, \forall x \in \mathcal{Y}(k), \forall k \in \mathcal{K}.$

Fact 2. $\|x\|_1 = \lfloor \frac{|k|}{2} \rfloor, \forall x \in co(\mathcal{Y}(k)), \forall k \in \mathcal{K}.$

Based on Facts 1 and 2, we present the next lemma that calculates the ℓ_1 norm of an arbitrary vector on the boundary

of the capacity region. See Section IX-A for the proof.

Lemma 1. *Let $\lambda_M \in \Lambda^{(M)}$. Then $\|\lambda_M\|_1 = \frac{Mp}{2} - \frac{1-(1-2p)^M}{4}$.*

The above lemma, proves that the vectors on the boundary of the capacity region $\Lambda^{(M)}$ have constant ℓ_1 norm that depends (solely) on the values of M and p. For the comparison of the capacities with respect to the total throughput achieved when we increase the number of quantum memories, we introduce the *capacity gain* as follows.

Definition 1 (Capacity Gain)**.** *Fix some rate vectors $\lambda_{M'} \in \Lambda^{(M')}$ and $\lambda_M \in \Lambda^{(M)}$. We define the capacity gain of adding $M' - M$ memories in a quantum switch with M memories as:*

$$g_{M\to M'} = \frac{\|\lambda_{M'}\|_1}{\|\lambda_M\|_1}.$$

Remark 1. *Due to Lemma 1 the capacity gain is independent of the rate vectors that we pick in the boundaries of the capacity regions and it depends only on the number of memories, M', M, and the probability p. For co-linear vectors i.e., $\lambda_{M'} = \alpha\lambda_M$ for $\alpha > 1$ (Figure 5(a)), note also that the choice of norm does not affect the definition of the capacity gain since for every norm it holds that $\frac{\|\lambda_{M'}\|_\cdot}{\|\lambda_M\|_\cdot} = \frac{\|\alpha\lambda_M\|_\cdot}{\|\lambda_M\|_\cdot} = \alpha = g_{M\to M'}$.*

Remark 2. *For probability of successful LLEs one, i.e., $p = 1$ and for even M and M', it holds that 1) $g_{M\to M+1} = 1$, and 2) $g_{M\to M'} = \frac{M'}{M}$.*

Therefore, in case the probability of LLE being successful is one there is not any point on having odd number of memories when requests are bipartite. From the second statement of Remark 2, we observe that when the LLEs always succeed, we get diminishing returns from adding memories, e.g., $g_{M\to M+2} = \frac{M+2}{M}$. This ratio indicates a linear increase of the "length" of the rate vector with respect to M.

In Figure 6a, we plot the capacity gain for various values of p with varying M, when we add one memory. As indicated from Remark 2, for high probability p there is not much incentives to have odd number of memories and, in general, we get diminishing returns from adding quantum memories for fixed p. That is, the gain converges to one for large Ms.

Figure 6b studies the ℓ_1 norm of an arbitrary rate vector $\lambda_M \in \Lambda^{(M)}$ given in Lemma 1. As indicated in Remark 2, for $p = 1$, the ℓ_1 norm increases linearly with even M. However, even for arbitrary values for p and M, the growth with respect to M is almost linear, since the non-linear term in Lemma 1 initially oscillates between 0 and 0.5 and for $p < 1$ it converges to 0. The closer p is to 0.5 the closer this term is to zero. For large M, the linear term in the ℓ_1 norm dominates the non-linear one, giving it the linear behavior shown in Figure 6b.

Figure 6c studies the ℓ_1 norm of an arbitrary rate vector $\lambda_M \in \Lambda^{(M)}$ given in Lemma 1 for varying probabilities p. Note, that the ℓ_1 norm increases linearly with respect to p but the slope of the line depends on M. Therefore, since the dominating part (for large M) of the formula given in Lemma 1 is given by $Mp/2$, to increase the capacity of the switch we have to *jointly* increase the number of memories and the probability of successfull LLEs. Figures 6b and 6c indicates that solely increasing M or p provides diminishing returns.

Conclusions: We prove that as we increase the number of memories in the switch the ℓ_1 norm of the rate vectors that we can achieve increases *almost* linearly with a slope that depends on the probability of successfull LLEs, p. Therefore, in order to increase the capacity of the switch we have to *jointly* increase the number of memories and the probability of successfull LLEs (quality of the quantum hardware/entanglement distillation). Solely increasing the number of memories or the quality of the links provides diminishing returns.

Algorithm 1 (MEW)

Set: $t = 0$
while switch is operating **do**
  $t \leftarrow t + 1$
  (S1) Quantum memory allocation: Select the memory allocation

$$m(t) \in \arg\max_{m\in\mathcal{M}} \sum_{r=1}^{R} Q_r(t)\mu_r(m, Q(t)), \tag{5}$$

  where

$$\mu(m, Q(t)) := \sum_{k\in\mathcal{K}(m)} \mathbb{P}(k)w(k, Q(t)) \tag{6}$$

$$w(k, Q(t)) \in \arg\max_{u\in\mathcal{B}(m,k)} \sum_{r=1}^{R} Q_r(t)u_r. \tag{7}$$

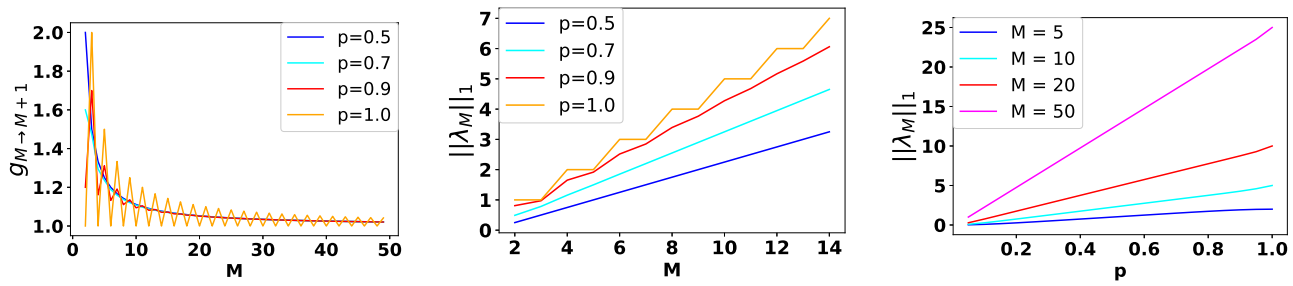
  (S2) LLEs generation: The switch attempts to create LLEs with the clients that have a memory connected. The successful LLEs determine the switch connectivity $k(t)$ and the action set $\mathcal{B}(m(t), k(t))$.
  (S3) Requests service: Select a service vector $b(t) \in \arg\max_{u\in\mathcal{B}(m(t),k(t))} \sum_{r=1}^{R} Q_r(t)u_r$
  (S4) Queue update: $Q(t+1) = [Q(t) - b(t)]^+ + A(t)$
end while

IV. MEW: A Throughput Optimal Policy

In this section we present a memory allocation policy that is throughput optimal. In Section IV-B, we discuss the scalability of the proposed policy.

A. MEW: A max-weight algorithm for allocating quantum memory and serving requests

We present *Maximum Expected Weight (MEW)*, an algorithm that stabilizes the queues when the arrival rate of requests is in the *interior* of the capacity region. MEW (Algorithm 1) consists of three steps. The first step **(S1)** allocates the quantum memories to clients using (5), which consists of maximizing the sum of the expected service in each queue (i.e., μ_r) multiplied by the queue occupancies (i.e., Q_r). This update can be regarded as an "expected" max-weight maximization, where the updates in (6) and (7) are intermediate steps to compute $\mu(m, Q(t)) = (\mu_1(m, Q(t)), \ldots, \mu_R(m, Q(t)))$ used in (5). The second step **(S2)** generates the LLEs with the clients



(a) (b) (c)

Figure 6. (a) The capacity gain, $g_{M\to M+1}$, for varying number of quantum memories, M. (b) The ℓ_1 norm of *any* arbitrary rate vector $\lambda_M \in \Lambda^{(M)}$ for varying M. (c) The ℓ_1 norm of *any* arbitrary rate vector $\lambda_M \in \Lambda^{(M)}$ for varying probability of LLE, p.

that have a memory connected. Only some LLE attempts succeed, which affects the network connectivity and the set of admissible requests service vectors, i.e., set $\mathcal{B}(m(t), k(t))$. The third step **(S3)** consists of finding the service vector $b(t)$ that maximizes the dot product with the vector of queues $Q(t)$. We have the following theorem. See Section IX-B for the proof.

Theorem 1. *Consider the quantum switch model in Section II, and suppose that the long-term arrival rate of requests λ is in the interior of the capacity region Λ. That is, there exists a vector $\hat{b} \in \Lambda$ such that $\lambda_r + \epsilon \leq \hat{b}_r, \forall r \in \{1, \dots, R\}$ for some $\epsilon > 0$. Then, MEW (Algorithm 1) ensures that the queues are strongly stable:*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{r=1}^{R} \mathbb{E}[Q_r(t)] \leq \frac{N^2}{\epsilon}.$$

Strong stability implies that all the requests that arrive are eventually served (i.e., (2) is satisfied), but also that the queues are bounded [31]. The result in Theorem 1 is based on max-weight techniques widely employed in network scheduling problems [30], [23], and the novelty of our contribution resides in the fact that the switch connectivity is random and depends on how we assign quantum memories to links/clients. The latter is different from wireless network models with time-varying connectivity since the allocation of quantum memories affects the switch's connections and, therefore, the set of admissible service vectors. Such coupling is typically not allowed in max-weight or backpressure approaches where the set of available actions can vary over time; however, usually in an i.i.d. manner [32]. In our problem, the action sets $\{\mathcal{B}(m(t), k(t))\}_{t=1}^{\infty}$ are not i.i.d. because they depend on the memory allocation decisions $\{m(t) \in \mathcal{M}\}_{t=1}^{\infty}$. Our approach to tackle this problem is to exploit the linearity of (5), (6), and (7), and evaluate all the possible scheduling decisions for every connectivity. However, enumerating all the cases can be computationally expensive sometimes, as we discuss next.

B. MEW scalability

The step with higher computational cost is the allocation of quantum memories **(S1)**, which involves computing (5), (6), and (7). In brief, the maximization in (5) is over the set of all possible memory allocations, which has cardinality $|\mathcal{M}| = \binom{N}{M}$.[7] Furthermore, we need to compute (6) and (7) for every memory allocation $m \in \mathcal{M}$ and network connectivity $k \in \mathcal{K}(m)$ respectively. We could compute (7) only once per switch connectivity since a switch connectivity can be obtained by different quantum memory allocations (see example in Figure 4). However, the number of possible switch connectivities increases exponentially with the number of memories since $|\mathcal{K}(m)| = 2^M$. In addition, the update in (7) requires finding a maximum-weighted matching in a complete hypergraph,[8] which is known to be an NP-hard problem [33].

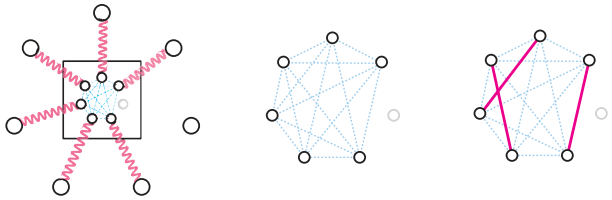


Figure 7. Quantum switch with $N = 7$ clients and $M = 6$ memories. When requests for end-to-end entanglements involve only two clients, the update in (7) reduces to finding a maximum weighted matching in a complete graph.

In sum, MEW does not scale well with N and M since it needs to solve, in the worst case, an exponential number of NP-hard problems for every memory allocation. Nevertheless, MEW can be used effectively when N, M and R are not very large. For instance, in Section VII, which includes our numerical evaluation, we execute MEW for settings with $N = 6, M = 3, R = 35$ and $N = 7, M = 4, R = 21$. In Section VII we study MEW when the update in (5) is carried out approximately. In particular, when MEW checks only l memory allocations out of all the $\binom{N}{M}$ possibilities. We refer to such algorithm as l-Approximate MEW.

In the next section, we focus on a special case where we can derive a variant of MEW (MEW2) that has polynomial complexity.

V. MEW2: Efficient scheduling in bipartite requests and successful LLEs.

In this section, we study the case where (i) LLE attempts are always successful (i.e., $p_n = 1, \forall n \in \{1, \dots, N\}$), and (ii)

[7] Assuming we allocate all the memories to clients.

[8] A hypergraph is a generalization of a graph in which an edge can connect any number of vertices.

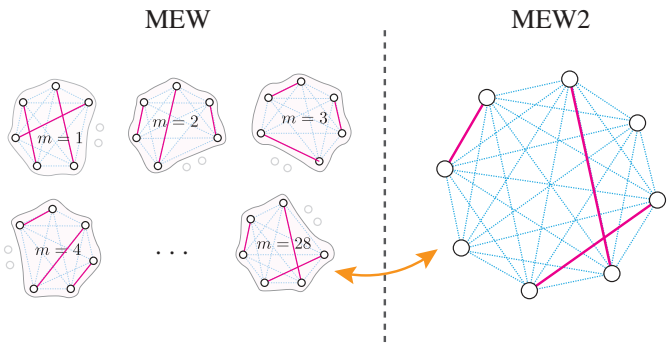


Figure 8. Schematic illustration of how MEW compares to MEW2 for a switch with $N=8$ clients and $M=6$ quantum memories. MEW has to find a maximum weighted matching in each of the $\binom{8}{6}=28$ different complete graphs (with 6 nodes each). MEW2 selects a special type of matching with at most $M/2$ edges in the 8-complete graph.

requests involve only two clients.[9] These cases are important for two reasons. First, we can appropriately extend the duration of a time slot to perform entanglement distillation and retrials for the entanglement attempts until we successfully generate all the LLEs in every time slot (see Section II-A).[10] Second, every multipartite request between clients can be divided into (multiple) bipartite ones. That is because two-qubit gates are universal, i.e., every quantum program can be implemented with two-qubit gates [11].

This case allows us to derive a variant of MEW (MEW2) that has lower computational cost. Specifically, we can allocate memories and select which requests to serve by finding a special type of matching with at most $M/2$ edges in an N-complete graph (see Algorithm 2).[11]

A. Motivation: Complexity of MEW

When end-to-end entanglement requests involve only two clients, (7) corresponds to finding a maximum weighted matching in the complete graph of the clients with an active LLE (see Figure 7). Finding such matching has polynomial time complexity [34]. The assumption that LLEs are always successful is useful to reduce the number of times we call (7). In particular, we have that a memory allocation is associated with a *single* switch connectivity. Hence, $|\mathcal{K}(m(t))|=1$ for all $m(t)\in\mathcal{M}$ and $\mu(m(t),Q(t))=w(k(t),Q(t))$ since $m(t)=k(t)$. In sum, we can solve (7) in polynomial time and only once for every admissible memory allocation. Yet, that can still be too much in some cases. For example, if $N=16$ and $M=8$, we need to find a maximum weighted matching of $\binom{16}{8}=12870$ different graphs to make a single memory allocation decision.

[9]Without loss of generality we assume that M is even. With bipartite requests, having an odd M means that there will be an unused memory.

[10]Note that there is a trade-off between reducing the duration of a time slot $\mathcal{T}$ (thus increasing the number of service requests per unit of time), and increasing the probability of successful LLEs.

[11]An N-complete graph is a graph with N nodes, in which each pair of vertices is connected with an edge. The edges' weights are the queue backlogs.

Algorithm 2 (MEW2)

1: **Set:** $t=0$
2: **while** switch is operating **do**
3: $t\leftarrow t+1$
4: **(S1b) Quantum memory allocation:** Select a matching with at most $M/2$ edges in the N-complete graph with maximum possible weight:

$$l(t)\in\arg\max_{u\in O}\sum_{r=1}^{R}Q_r(t)\;u_r, \tag{8}$$

where $O:=\{u\in\mathcal{P}_N:\sum_{r=1}^{R}u_r\leq M/2\}$ and $\mathcal{P}_N$ the set of matchings in the N-complete graph. Assign a memory to every client/node that is connected to an edge in $l(t)$, i.e.,

$$m(t)\in\{m\in\mathcal{M}:l(t)\in\mathcal{P}(\mathcal{C}(m))\}, \tag{9}$$

where $\mathcal{C}(m)$ is the complete graph of the clients $n\in\{1,\ldots,N\}$ with $m_n=1$.
5: **(S2b) LLE generation:** Generate LLEs with the clients that have a memory connected.
6: **(S3b) Requests service:** Select $b(t)=l(t)$
7: **(S4b) Queue update:** $Q(t+1)=[Q(t)-b(t)]^{+}+A(t)$
8: **end while**

B. Maximum Expected Weight 2 (MEW2)

We propose *Maximum Expected Weight 2 (MEW2)*, a policy that selects a memory allocation by obtaining a special type of matching in the N-complete graph (Algorithm 2). The intuition behind MEW2 is shown in Figure 8 for a switch with $N=8$ and $M=6$. Recall that **(S1)** in MEW computes a maximum weighted matching in $\binom{N}{M}$ different M-complete graphs, and picks one with maximum weight. Observe from the figure that such matching is also a (non-maximal) matching in the N-complete graph. Thus, we can replace step **(S1)** in MEW by *directly* computing a matching with maximum weight among the matchings that have at most $M/2$ edges. We have the following corollary of Theorem 1 which is proved in the Appendix (Section IX-C).

Corollary 1 (Theorem 1)**.** *Consider the setup of Theorem 1 where the LLE attempts are always successful (i.e., $p_n=1,\forall n\in\{1,\ldots,N\}$). Also, suppose that requests involve connecting two clients and that M is even. Then, MEW2 ensures that the queues are strongly stable.*

Finding the matching described in (8), which characterizes the complexity of MEW2, can be done in polynomial time. In particular, we can find such matching by augmenting the N-complete graph and then computing a maximum weighted matching. Specifically, the augmented graph has $n-M$ virtual nodes connected to the others with edges that have *infinite* weight. The solution to (8) corresponds to a maximum weighted matching of the augmented graph, which can be found in polynomial time [35], [34].

VI. Relaxation of Assumption 1 - Multiple Memories & Channels

For ease of exposition of the mathematical model and its optimization, we assumed that that we have only one channel per client and we can assign one memory to each channel (Assumption 1). Nevertheless, our model and algorithms can be extended accordingly to account for the case of a quantum switch with multiple memories ($M > N$) and multiple channels per client (> 1).

In Figure 9 we focus on the case of one channel per client and multiple memories associated with it. One can generalize this to the case of multiple channels per client. What changes from the case of one memory per client (Section II-A) is that the client can now emit a photon (say every ω seconds) that is entangled with an electron spin inside a different memory every time. Therefore, at $\tau = 0$ one photonic qubit can be emitted through the channel and at $\tau = \omega$ a second one. At time $\tau = m\omega$, where m is the number of memories inside the client, all of the photonic qubits are emitted and no more can be emitted in case we can utilize only one nuclear-spin in every memory. At time $\tau = L/c$, as in (Section II-A) the first photonic qubit reaches the switch and the switch "stores" its state to a nuclear-spin inside a memory allocated to the client. This procedure might fail but the outcome should be heralded with a classical bit that arrives at the client at time $\tau = 2L/c$. However, in this case we have multiple photonic qubits inside the channel that arrives at the switch in the time interval $2L/c \leq \tau \leq 2L/c + m\omega$. At $\tau = 2L/c + m\omega$ every nuclear-spin qubit might have been entangled with a nuclear-spin qubit inside the clients' memories. The switch can then perform entanglement distillation using the successful LLEs. For the unused memories, the switch will retry generating LLEs until $\tau = \Delta$ when the switch performs BSMs to satisfy the requests for end-to-end entanglement.

In this level of abstraction, we do not assume a specific distillation protocol and we only care about the probability that each possible connectivity appears at time $\tau = \Delta$. The model just needs the probability for successful LLEs (of sufficient fidelity) for every memory at time $\tau = \Delta$. Such a model can express any distillation scheme since depending on how many entanglements are fused to create one with greater fidelity, the probabilities of successful LLEs at time $\tau = \Delta$ can be changed accordingly. For example, if we employ an m to 1 distillation scheme, there is only one non zero probability of successful LLE at $\tau = \Delta$ for every m memories. The physical model described in this section for the case of multiple channels/memories per client, can be again abstracted by the same mathematical model that we proposed in this paper by just adding multiple memory allocation possibilities depending on the setting. Therefore, we can incorporate multiple memories and multiple channels per client by appropriately extending the action set.

VII. Numerical Evaluation

In this section, we illustrate the behavior of MEW and MEW2 in different scenarios (e.g., request load, LLEs generation). Moreover, we study MEW when the update in (5) is carried out approximately. In particular, when MEW uses only l memory allocations out of all the $\binom{N}{M}$ possibilities (Sections VII-B and VII-C). We refer to such algorithm as l-Approximate MEW.

A. Simulation 1: Performance of MEW under different arrival rates

This simulation evaluates the performance of MEW when the requests arrive with three different intensities: 70%, 99%, and 120% of the total load that the switch can support.[12] For the simulation, we set $N = 6$, $M = 3$, and $R = 8$, where all the requests involve connecting three clients (i.e., the requests are tripartite). The probability of the LLE attempts being successful is fixed to $p_n = 0.9$ for all clients and loads.

We run MEW for the three different loads and show the evolution of the queue occupancies over time in Figure 10. Observe from the figure that when the arrival rates are in the interior of the capacity region (70% and 99%), the backlogs remain bounded. However, the "saturation" points are different, which is in line with the queue stability bound in Theorem 1. Higher intensity (i.e., smaller ϵ in Theorem 1) implies larger backlogs. Finally, when the request arrival intensity is equal to 120% (outside of the capacity region), the queues are not stable since their occupancy increases linearly.

Conclusions: MEW stabilizes the queues when the arrivals are in the interior of the capacity region. The average queue occupancies depend on how close the long-term arrival rates are to the boundary of the switch's capacity (Theorem 1).

B. Simulation 2: MEW with memory allocation decision deadlines

Recall from Section IV-B that MEW needs to solve (7) multiple times for every memory allocation decision. However, we may not be able to evaluate all possible memory allocations since in practice we need to select one within a time deadline. To capture that, we reduce the search space of problem (5).

As in Section VII-A, we consider $N = 6$, $M = 3$, and that the probability of LLE attempts being successful equals $p_n = 0.9$ for all clients. However, we now consider all types of bipartite and tripartite requests (i.e., $R = \binom{6}{2} + \binom{6}{3} = 35$), hence the time needed to compute (7) increases.

We run MEW and the $\{1, 10\}$-Approximate variant for different arrival rate intensities (70%, 99%, and 120%) and show the results in Figure 11. Observe from the figure that MEW stabilizes the queues when the arrivals are in the interior of the capacity region. However, for the l-approximation, the stability depends on the value of l and the traffic intensity. Specifically, the 1-Approximate MEW stabilizes the system when the traffic intensity is 70% (Figure 11a), but not when the intensity is equal to 99% (Figure 11b). In contrast, the 10-Approximation keeps the queues bounded similar to MEW.

Conclusions: The l-Approximate MEW can stabilize the queues when the value of l is large enough. How large l should be is related to how close the arrival rates are to the boundary of the capacity region. As future work, it is interesting to

[12] An intensity of 100% is at the boundary of the capacity region.

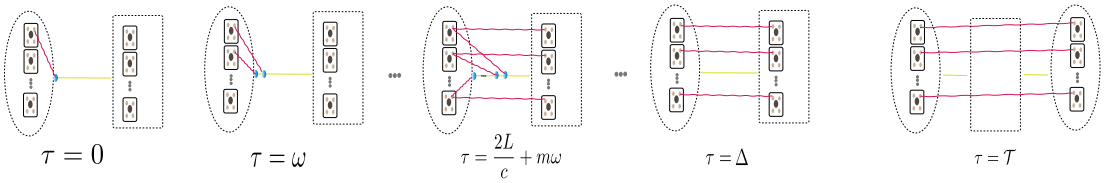


Figure 9. Physical level operation of a quantum switch with multiple quantum memories dedicated to a client's channel, during a time slot. With appropriate changes to MEW and MEW2, the technical work of this paper can be extended appropriately to include multiple memories and multiple channels per client.

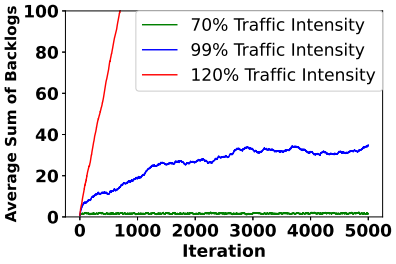


Figure 10. Illustrating the simulation in Section VII-A: The evolution of MEW for a quantum switch with 6 clients and 3 memories for different arrival rates. LLE attempts succeed with probability 0.9. Each line is the average of 10 different realizations.

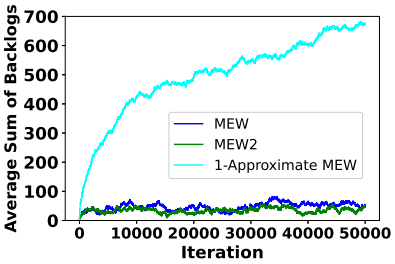
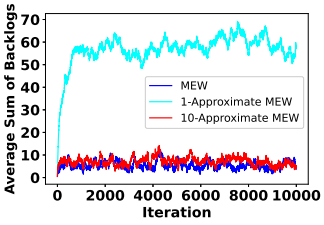
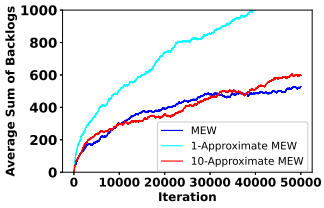


Figure 12. Illustrating the evolution of MEW, MEW2 and 1-Approximate MEW for a quantum switch with $N = 7$, $M = 4$ and bipartite requests. The traffic intensity is 99% and the LLE attempts succeed with probability 0.9. Each line is the average of 10 different realizations.

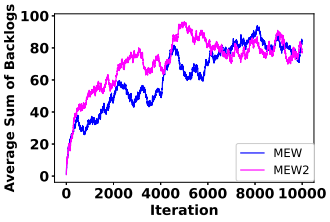


(a) 70% Traffic intensity

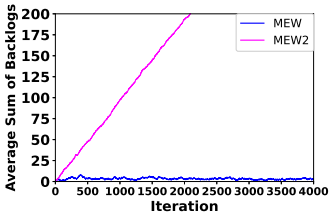


(b) 99% Traffic intensity

Figure 11. Illustrating the simulation in Section VII-B: The evolution of MEW and $\{1, 10\}$-Approximate MEW for a quantum switch with $N = 6$, $M = 3$ and $p_n = 0.9$ for every client n. Each line is the average of 10 different realizations.



(a)



(b)

Figure 13. Illustrating the simulation in Section VII-D: The evolution of MEW2 and MEW for a quantum switch with $N = 7$, $M = 4$, requests for bipartite end-to-end entanglement and (a) homogeneous probabilities for successful LLEs, $p_n = 0.7$, and, (b) heterogeneous probabilities for successful LLEs, $(p_1, p_2, \ldots, p_7) = (0.3, 0.4, \ldots, 0.9)$. Each line is the average of 10 different realizations.

investigate how the capacity region scales when the memory allocation is obtained approximately (e.g., as a function of the parameter l).

C. Simulation 3: Performance of MEW2

In this simulation, we compare MEW2 to MEW and 1-Approximate MEW in a switch with $N = 7$ clients and $M = 4$ memories. The traffic intensity is fixed to 99%, and we assume that the LLE attempts are always successful. We consider bipartite requests with $R = \binom{7}{2} = 21$.

We run the three algorithms and show the results in Figure 12. Observe that MEW2 can keep the queues stable and that its behavior is similar to MEW. Nonetheless, recall that MEW has as higher computational cost than MEW2 (see discussion in Section V). Next, observe from Figure 12 that the 1-Approximate MEW (which has a comparable cost to MEW2) cannot stabilize the queues, which is similar to the result in the previous experiment when the traffic intensity is 99%. Recall that MEW2 requires that the LLE attempts are always successful and that the entanglement requests are bipartite.

Conclusions: The behavior of MEW2 is similar to MEW even though its complexity is significantly lower. The 1-Approximate MEW does not keep the queues stable despite having a similar computational cost to MEW2.

D. Simulation 4: MEW2 when the LLE attempts are not always successful

MEW2 can stabilize the queues when the LLE attempts are always successful. However, that may not be the case in all practical systems. In this section, we study the robustness of MEW2 when the LLE attempts can fail. Since MEW2 assumes that $p_n = 1$, we allow it to perform one last maximum weight operation to decide which requests to serve depending on the switch connectivity (in analogy to step (**S3**) in MEW).

We consider a switch with $N = 7$ clients and $M = 4$ memories, and study the case where the LLE attempts probabilities are (i) homogeneous with $p_n = 0.7$ for all $n \in \{1, \ldots, N\}$, and (ii) heterogeneous with $(p_1, p_2, \ldots, p_7) = (0.3, 0.4, \ldots, 0.9)$. Also, as in the previous section, the requests are bipartite and the traffic intensity is equal to 99%.

We run MEW2 and show the evolution of the queues over time in Figure 13. Observe that in the homogeneous case

(Figure 13a), MEW2 stabilizes the queues even though it does not know the probabilities with which LLE attempts succeed. However, when the probabilities are heterogeneous (Figure 13b), MEW2 is not able to stabilize the queues. We conjecture that MEW2 behaves better in the homogeneous case since the latter is closer to the case where every LLE attempt succeed with probability one (i.e., the assumption in Corollary 1).

Conclusions: When the probabilities p_n are homogeneous throughout the clients, we observe that MEW2 stabilizes the system even though p_n are unknown.

VIII. Conclusions

In this paper, we have studied the problem of operating a quantum switch with memory constraints. The switch has to allocate a limited number of quantum memories to clients to generate link-level entanglements (LLEs), and then use these to serve end-to-end entanglements requests. Our model is different from existing approaches since we capture that quantum memory is a scarce resource (in analogy to quantum computers). The paper's main contribution is threefold: (i) to characterize the switch's capacity region, (ii) to study how the capacity scales with respect to the number of quantum memories and probability of LLE success and (iii) to propose a memory allocation policy (MEW) that is throughput optimal. In addition, we present MEW2, a variant of MEW tailored to the case where end-to-end entanglements are bipartite and LLE attempts are always successful. MEW2 finds a memory allocation by obtaining a special type of matching in the N-complete graph in polynomial time.

IX. Appendix

A. Proof of Lemma 1

The proof is based on expanding the vector λ_M using (4) as follows:

$$\begin{aligned}
&\|\lambda_M\|_1 = \\
&= 1^T \Big(\sum_{m\in\mathcal{M}^{(M)}} \theta_m \sum_{k\in\mathcal{K}(m)} p^{|k|}(1-p)^{M-|k|}\ x^{m,k}\Big) \\
&= \sum_{m\in\mathcal{M}^{(M)}} \theta_m \sum_{k\in\mathcal{K}(m)} p^{|k|}(1-p)^{M-|k|}\ \lfloor\frac{|k|}{2}\rfloor \\
&= \sum_{m\in\mathcal{M}^{(M)}} \theta_m \sum_{\kappa\in\{1,\ldots,M\}} \binom{M}{\kappa} p^{\kappa}(1-p)^{M-\kappa}\ \lfloor\frac{\kappa}{2}\rfloor \\
&= \sum_{\kappa\in\{1,\ldots,M\}} \binom{M}{\kappa} p^{\kappa}(1-p)^{M-\kappa}\ \lfloor\frac{\kappa}{2}\rfloor \\
&= \sum_{\kappa\in\{1,\ldots,M\}:kmod2=1} \binom{M}{\kappa} p^{\kappa}(1-p)^{M-\kappa}\frac{(\kappa-1)}{2} \\
&\quad + \sum_{\kappa\in\{1,\ldots,M\}:kmod2=1} \binom{M}{\kappa} p^{\kappa}(1-p)^{M-\kappa}\frac{\kappa}{2} \\
&= \sum_{\kappa\in\{1,\ldots,M\}} \binom{M}{\kappa} p^{\kappa}(1-p)^{M-\kappa}\ \frac{\kappa}{2} \\
&\quad - \sum_{\kappa\in\{1,\ldots,M\}:kmod2=1} \binom{M}{\kappa} p^{\kappa}(1-p)^{M-\kappa}\frac{1}{2} \\
&= \frac{Mp}{2} - \frac{1-(1-2p)^M}{4}.
\end{aligned}$$

B. Proof of Theorem 1

We prove that the queues are stable by using a quadratic Lyapunov function, and ultimately showing that the proposed policy has expected negative drift. That is, the queues at time slots t and $t+1$ satisfy: $\mathbb{E}[\|Q(t+1)\|^2 - \|Q(t)\|^2] < 0$, where the expectations are w.r.t. the (i) request arrivals; (ii) successful LLEs; and (iii) all possible queue values at time t. To start, observe that $\|Q(t+1)\|^2 = \|[Q(t)-b(t)]^+ + A(t)\|^2 = \|[Q(t)-b(t)]^+\|^2 + \|A(t)\|^2 + 2\sum_{r=1}^R [Q_r(t)-b_r(t)]^+ A_r(t) \le \|Q(t)-b(t)\|^2 + \|A(t)\|^2 + 2\sum_{r=1}^R [Q_r(t)-b_r(t)]^+ A_r(t) = \|Q(t)\|^2 + \|b(t)\|^2 + \|A(t)\|^2 + 2\sum_{r=1}^R [Q_r(t)-b_r(t)]^+ A_r(t) - 2\sum_{r=1}^R Q_r(t)b_r(t)$, where the inequality follows since $\|[v]^+\|^2 \le \|v\|^2$ for any vector $v \in \mathbb{R}^n$. Also, we have $-2\sum_{r=1}^R Q_r(t)b_r(t) + 2\sum_{r=1}^R [Q_r(t)-b_r(t)]^+ A_r(t) \le -2\sum_{r=1}^R Q_r(t)b_r(t) + 2\sum_{r=1}^R Q_r(t)A_r(t) = 2\sum_{r=1}^R Q_r(t)(A_r(t)-b_r(t))$, where the inequality follows since $-\sum_{r=1}^R b_t(t)A_r(t)$ is non-positive.

Next, since $\|A(t)\|^2 \le N^2$, $\|b(t)\|^2 \le N^2$ (by assumption), and $\mathbb{E}[A(t)] = \lambda$ by assumption, we can take expectations with respect to $A_r(t)$ for a fixed queue $Q(t)$ to obtain:

$$\begin{aligned}
&\mathbb{E}[\|Q(t+1)\|^2 - \|Q(t)\|^2 \mid Q(t)] \\
&\qquad \le 2N^2 + 2\sum_{r=1}^R Q_r(t)(\lambda_r - b_r(t)).
\end{aligned} \tag{10}$$

We proceed to upper bound the expected value of $-\sum_{r=1}^R Q_r(t)b_r(t)$. Since $b(t)$ is a random vector that depends on the switch connectivity and memory allocation at time t, we have $\mathbb{E}[-\sum_{r=1}^R Q_r(t)b_r(t)] = -\sum_{r=1}^R Q_r(t)\mu_r(m(t),Q(t))$, where $\mu_r(m(t),Q(t))$ is defined in (6). Note that $Q(t)$ does not depend on the switch connectivity in time slot t. Combining the last equation with (10), we have

$$\begin{aligned}
&\mathbb{E}[\|Q(t+1)\|^2 - \|Q(t)\|^2 \mid Q(t)] \\
&\qquad \le 2N^2 + 2\sum_{r=1}^R Q_r(t)(\lambda_r - \mu_r(m(t),Q(t))),
\end{aligned} \tag{11}$$

where the expectation is with respect to the switch connectivities for a fixed memory allocation.

Next, observe that the memory allocation rule in (5) ensures that:

$$\begin{aligned}
&-\sum_{r=1}^R Q_r(t)\mu_r(m(t),Q(t)) \\
&\qquad \le -\sum_{r=1}^R Q_r(t)\mu_r(m,Q(t)) \qquad \forall m\in\mathcal{M}.
\end{aligned} \tag{12}$$

since $\mu(m(t),Q(t))$ maximizes $\sum_{r=1}^R Q_r(t)\mu_r(m(t),Q(t))$.

Now, let $\theta_m \geq 0$ with $\sum_{m\in\mathcal{M}} \theta_m = 1$ and observe that

$$
\begin{aligned}
&-\sum_{r=1}^{R} Q_r(t)\mu_r(m(t),Q(t)) \\
&\quad = -\sum_{m\in\mathcal{M}} \theta_m \sum_{r=1}^{R} Q_r(t)\mu_r(m(t),Q(t)) \\
&\quad \overset{(a)}{\leq} -\sum_{m\in\mathcal{M}} \theta_m \sum_{r=1}^{R} Q_r(t)\mu_r(m,Q(t)), \\
&\quad \overset{(b)}{=} -\sum_{m\in\mathcal{M}} \theta_m \sum_{r=1}^{R} Q_r(t) \sum_{k\in\mathcal{K}(m)} \mathbb{P}(k;m) w_r(k,Q(t)) \\
&\quad \overset{(c)}{=} -\sum_{m\in\mathcal{M}} \theta_m \sum_{k\in\mathcal{K}(m)} \mathbb{P}(k;m) \sum_{r=1}^{R} Q_r(t) w_r(k,Q(t)),
\end{aligned}
$$

where (a) follows by (12), (b) by (6), and (c) by rearranging terms.

Now, recall $w(k,Q(t))$ maximizes $\sum_{r=1}^{R} Q_r(t)w_r(k,Q(t))$ because how we defined it in (7), and let $\delta_b^{m,k} \geq 0$ for all $b \in \mathcal{B}(m,k)$ with $\sum_{b\in\mathcal{B}(m,k)} \delta_b^{m,k} = 1$ for every $k \in \mathcal{K}(m)$ and $m \in \mathcal{M}$. As before, we have

$$
\begin{aligned}
&-\sum_{r=1}^{R} Q_r(t)w_r(k,Q(t)) \qquad\qquad k \in \mathcal{K}(m) \\
&\quad = -\sum_{b\in\mathcal{B}(m,k)} \delta_b^{m,k} \sum_{r=1}^{R} Q_r(t)w_r(k,Q(t)) \\
&\quad \overset{(a)}{\leq} -\sum_{b\in\mathcal{B}(m,k)} \delta_b^{m,k} \sum_{r=1}^{R} Q_r(t)b_r \\
&\quad = -\sum_{r=1}^{R} Q_r(t) \sum_{b\in\mathcal{B}(m,k)} \delta_b^{m,k} b_r,
\end{aligned}
$$

where b in inequality (a) holds for any vector in $\mathcal{B}(m,k)$. Combining the previous equations, $-\sum_{r=1}^{R} Q_r(t)\mu_r(m(t),Q(t)) \leq -\sum_{r=1}^{R} Q_r(t)\hat{b}_r$, where $\hat{b} = \sum_{m\in\mathcal{M}} \theta_m \sum_{k\in\mathcal{K}(m)} \mathbb{P}(k;m) \sum_{b\in\mathcal{B}(m,k)} \delta_b^{m,k} b$ is any vector in Λ (Proposition 1). Hence, from inequality (11), we have $\mathbb{E}[\|Q(t+1)\|^2 - \|Q(t)\|^2 \mid Q(t)] \leq 2N^2 + 2\sum_{r=1}^{R} Q_r(t)(\lambda_r - \hat{b}_r(t))$. The rest of the proof follows the usual max-weight arguments. Because $\lambda_r + \epsilon \leq \hat{b}_r$ for some $\epsilon > 0$ by assumption, it holds $\mathbb{E}[\|Q(t+1)\|^2 - \|Q(t)\|^2 \mid Q(t)] \leq 2N^2 - 2\sum_{r=1}^{R} Q_r(t)\epsilon$. Now, take expectations w.r.t. all the possible values of $Q(t)$ to obtain

$$
\mathbb{E}[\|Q(t+1)\|^2 - \|Q(t)\|^2] \leq 2N^2 - 2\sum_{r=1}^{R} Q_r(t)\epsilon, \tag{13}
$$

and observe that $\mathbb{E}[\|Q(t+1)\|^2 - \|Q(t)\|^2] < 0$ when $\sum_{r=1}^{R} Q_r(t) > \frac{2N^2}{2\epsilon}$. That is, the queue drift is negative. Finally, sum (13) from $t = 1,\ldots,T$ to obtain $\mathbb{E}[\|Q(T+1)\|^2] - \mathbb{E}[\|Q(0)\|^2] \leq 2TN^2 - 2\epsilon\sum_{t=1}^{T}\sum_{r=1}^{R}\mathbb{E}[Q_r(t)]$. Rearranging terms and dividing by T yields $\frac{1}{T}\sum_{t=1}^{T}\sum_{r=1}^{R}\mathbb{E}[Q_r(t)] \leq \frac{N^2}{\epsilon} + \frac{\|Q(0)\|^2}{2T\epsilon}$, and taking $T \to \infty$ we obtain the stated result.

C. Proof of Corollary 1

Let $\mathcal{C}(m)$ be the complete graph that results from the clients $n \in \{1,\ldots,N\}$ for which $m_n = 1$. Moreover, let $\mathcal{P}(G)$ be the set of matchings of a graph G, and $\mathcal{P}_N$ be the set of matchings of the complete graph with N clients. The proof of Corollary 1 relies on finding a policy that is equivalent to MEW, which we proved in Theorem 1 that stabilizes the queues.

We can rewrite step **(S1)** of MEW, in a single line, as follows:

$$
m(t) \in \arg\max_{m\in\mathcal{M}} \max_{u\in\mathcal{P}(\mathcal{C}(m))} \sum_{r=1}^{R} Q_r(t)\ u_r. \tag{14}
$$

Problem (14) summarizes MEW when LLE attempts are always successful and the requests are bipartite. That holds true because (i) its solution picks the memory allocation of **(S1)**, and (ii) the maximizer of the inner optimization problem is the final service vector of step **(S3)**.

Let $M^*(t)$ be the set of solutions to problem (14) and define the sets $\Pi := \bigcup_{m\in\mathcal{M}} \mathcal{P}(\mathcal{C}(m))$, $O := \{u \in \mathcal{P}_N : \sum_{r=1}^{R} u_r \leq M/2\}$. Recall that we assume that M is even. Then, step **(S3)** in this special case, is equivalent to

$$
\begin{aligned}
b(t) &\in \bigcup_{m^*\in M^*(t)} \arg\max_{u\in\mathcal{P}(\mathcal{C}(m^*))} \sum_{r=1}^{R} Q_r(t)u_r \qquad (15)\\
&\overset{(a)}{=} \arg\max_{u\in\Pi} \sum_{r=1}^{R} Q_r(t)u_r \\
&\overset{(b)}{=} \arg\max_{u\in O} \sum_{r=1}^{R} Q_r(t)u_r. \qquad (16)
\end{aligned}
$$

In the above, equality (a) holds since we augment the search space of the optimization problem with matchings that do not maximize the inner product with the queue backlogs. Equality (b) holds because although Π and O contain matchings of seemingly different graphs, their nodes corresponds to the switch's clients and hence $O = \Pi$. Therefore, step **(S3)** of MEW can be carried out by solving the problem (16) (see step **(S1b)** in MEW2).

Panagiotis Promponas (Graduate Student Member, IEEE) received his Diploma in Electrical and Computer Engineering (ECE) from the National Technical University of Athens (NTUA), Greece, in 2019. He is currently a Ph.D. student in the Department of Electrical Engineering at Yale University. His primary scientific interests include resource allocation in constrained interdependent systems and the optimization of algorithms. Specifically, he focuses on the optimization and modeling of quantum networks and wireless networking systems. He is also the recipient (co-author) of the Best Paper Award at the 12th IFIP WMNC 2019.

Víctor Valls graduated with a degree in electrical engineering from Universitat Pompeu Fabra in 2011 and obtained his MSc from the same university in 2013. In 2017, he completed his Ph.D. in applied mathematics at Trinity College Dublin, Ireland. From 2019 to 2022, he was as a postdoctoral research fellow at Yale University (USA) supported by a Marie Skłodowska-Curie fellowship. Currently, he is a Staff Research Scientist at IBM Research Europe – Dublin. His research interests are in optimization, networks, and quantum computing.

Saikat Guha (Senior Member, IEEE) received the B.Tech. degree in electrical engineering in 2002 from the Indian Institute of Technology Kanpur, Kanpur, India, and the S.M. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2004 and 2008, respectively. He is currently the Peyghambarian Endowed Chair Professor of Optical Sciences with the University of Arizona, Tucson, AZ, USA, and the Director of the Center for Quantum Networks, an Engineering Research Center awarded by the NSF in 2020. He was previously a Lead Scientist with the Quantum Information Processing Group, Raytheon BBN Technologies, Cambridge. His research interests include quantum optics and information theory, with applications to quantum limits to optical communication and sensing, photonic quantum computing, and network information theory. In 1998, he was a Member of the first Indian Team with the International Physics Olympiad, where he was awarded the European Physical Society Award for the experimental component.

Leandros Tassiulas (Fellow, IEEE) is the John C. Malone Professor of Electrical Engineering at Yale University, where he served as department head 2016-2022. His current research is on intelligent services and architectures at the edge of next generation networks including Internet of Things, sensing actuation in terrestrial and non terrestrial environments and quantum networks. He worked in the field of computer and communication networks with emphasis on fundamental mathematical models and algorithms of complex networks, wireless systems and sensor networks. His most notable contributions include the max-weight scheduling algorithm and the back-pressure network control policy, opportunistic scheduling in wireless, the maximum lifetime approach for wireless network energy management, and the consideration of joint access control and antenna transmission management in multiple antenna wireless systems. Dr. Tassiulas is a Fellow of IEEE (2007) and of ACM (2020) as well as a member of Academia Europaea (2023). His research has been recognized by several awards including the IEEE Koji Kobayashi computer and communications award (2016), the ACM SIGMETRICS achievement award 2020, the inaugural INFOCOM 2007 Achievement Award "for fundamental contributions to resource allocation in communication networks," several best paper awards including the INFOCOM 1994, 2017 and Mobihoc 2016, a National Science Foundation (NSF) Research Initiation Award (1992), an NSF CAREER Award (1995), an Office of Naval Research Young Investigator Award (1997) and a Bodossaki Foundation award (1999). He holds a Ph.D. in Electrical Engineering from the University of Maryland, College Park (1991) and a Diploma of Electrical Engineering from Aristotle University of Thessaloniki, Greece. He has held faculty positions at Polytechnic University, New York, University of Maryland, College Park and University of Thessaly, Greece.

REFERENCES

[1] P. Promponas, V. Valls, and L. Tassiulas, "Full exploitation of limited memory in quantum entanglement switching," in *2023 IFIP Networking Conference (IFIP Networking)*, 2023, pp. 1–9.

[2] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[3] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[4] Ibm unveils new roadmap to practical quantum computing era; plans to deliver 4,000+ qubit system. [Online]. Available: https://newsroom.ibm.com/2022-05-10-IBM-Unveils-New-Roadmap-to-Practical-Quantum-Computing-Era-Plans-to-Deliver-4,000-Qubit-System

[5] "Ibm unveils breakthrough 127-qubit quantum processor," Available at https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor.

[6] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, Barends *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[7] R. Van Meter and S. J. Devitt, "The path to scalable distributed quantum computing," *Computer*, vol. 49, no. 9, pp. 31–42.

[8] S. Guha and C. Gagatsos, "Cluster-state quantum computing methods and systems," Jul. 7 2022, uS Patent App. 17/594,874.

[9] C. Qiao, Y. Zhao, G. Zhao, and H. Xu, "Quantum data networking for distributed quantum computing: Opportunities and challenges," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops*. IEEE, 2022, pp. 1–6.

[10] A. Yimsiriwattana and S. J. Lomonaco Jr, "Generalized ghz states and distributed quantum computing," *arXiv preprint quant-ph/0402148*, 2004.

[11] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[12] J.-W. Pan, D. Bouwmeester, H. Weinfurter, and A. Zeilinger, "Experimental entanglement swapping: entangling photons that never interacted," *Physical review letters*, vol. 80, no. 18, p. 3891, 1998.

[13] R.-B. Jin, M. Takeoka, U. Takagi, R. Shimizu, and M. Sasaki, "Highly efficient entanglement swapping and teleportation at telecom wavelength," *Scientific reports*, vol. 5, no. 1, pp. 1–7, 2015.

[14] W. Dai and D. Towsley, "Entanglement swapping for repeater chains with finite memory sizes," 2021. [Online]. Available: https://arxiv.org/abs/2111.10994

[15] C. Cicconetti, M. Conti, and A. Passarella, "Request scheduling in quantum networks," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 2–17, 2021.

[16] L. Le and T. N. Nguyen, "Dqra: Deep quantum routing agent for entanglement routing in quantum networks," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–12, 2022.

[17] N. K. Panigrahy, P. Dhara, D. Towsley, S. Guha, and L. Tassiulas, "Optimal entanglement distribution using satellite based quantum networks," *arXiv preprint arXiv:2205.12354*, 2022.

[18] G. Vardoyan, S. Guha, P. Nain, and D. Towsley, "On the exact analysis of an idealized quantum switch," *SIGMETRICS Perform. Eval. Rev.*, vol. 48, no. 3, pp. 79–80, mar 2021. [Online]. Available: https://doi.org/10.1145/3453953.3453971

[19] P. Nain, G. Vardoyan, S. Guha, and D. Towsley, "Analysis of a tripartite entanglement distribution switch," *Queueing Systems*, 2022.

[20] W. Dai, A. Rinaldi, and D. Towsley, "Entanglement swapping in quantum switches: Protocol design and stability analysis," 2021. [Online]. Available: https://arxiv.org/abs/2110.04116

[21] T. Vasantam and D. Towsley, "A throughput optimal scheduling policy for a quantum switch," in *Quantum Computing, Communication, and Simulation II*, P. R. Hemmer and A. L. Migdall, Eds. SPIE, mar 2022.

[22] G. Vardoyan, S. Guha, P. Nain, and D. Towsley, "On the stochastic analysis of a quantum entanglement switch," *ACM SIGMETRICS Performance Evaluation Review*, vol. 47, no. 2, pp. 27–29, 2019.

[23] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, 2006, vol. 1, no. 1.

[24] J.-W. Pan, C. Simon, Č. Brukner, and A. Zeilinger, "Entanglement purification for quantum communication," *Nature*, vol. 410, no. 6832, pp. 1067–1070, 2001.

[25] D. Levonian, R. Riedinger, B. Machielse, E. Knall, M. Bhaskar, C. Knaut, R. Bekenstein, H. Park, M. Lončar, and M. Lukin, "Optical entanglement of distinguishable quantum emitters," *Physical Review Letters*, vol. 128, no. 21, p. 213602, 2022.

[26] A. M. Tyryshkin, S. Tojo, J. J. Morton, H. Riemann, Abrosimov *et al.*, "Electron spin coherence exceeding seconds in high-purity silicon," *Nature materials*, vol. 11, no. 2, pp. 143–147, 2012.
[27] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress *et al.*, "Heralded entanglement between solid-state qubits separated by three metres," *Nature*.
[28] M. Mobayenjarihani, G. Vardoyan, and D. Towsley, "Optimistic entanglement purification with few quantum memories," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2021, pp. 439–440.
[29] G. Bacciagaluppi, "The Role of Decoherence in Quantum Mechanics," in *The Stanford Encyclopedia of Philosophy*, Fall 2020 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2020.
[30] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.
[31] M. J. Neely, "Stability and capacity regions or discrete time queueing networks," *arXiv preprint arXiv:1003.3396*, 2010.
[32] L. Tassiulas, "Scheduling and performance limits of networks with constantly changing topology," *IEEE Transactions on Information Theory*, vol. 43, no. 3, pp. 1067–1073, 1997.
[33] J. Han and A. Treglown, "The complexity of perfect matchings and packings in dense hypergraphs," 2016. [Online]. Available: https://arxiv.org/abs/1609.06147
[34] A. Schrijver *et al.*, *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003, vol. 24.
[35] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, pp. 449–467, 1965.