Entanglement Generation Protocol: Notes on Link+Physical Layer

Stephanie, Axel, Matthew, Leon, Erwin, Ronald

October 27, 2018

The objective of this document is to define the link layer in quantum networks connecting quantum processing nodes, and to propose concrete link layer protocols based on an existing implementation of the physical layer with certain properties. In analogy to classical networks, the objective of the link layer will be to enable communication between two nodes A and B connected by a *link* on the same network. Here, enabling communication corresponds to producing entanglement between A and B, and we will hence refer to such protocols as Entanglement Generation Protocols (EGP). We propose the desired service, interface to the higher layer, as well as two possible EGPs that are closely related design alternatives. We first discuss an EGP between two nodes A and B, and discuss extensions to a proposed architecture connecting many nodes at the end.

To fit the link layer EGP into the future envisioned network stack we briefly sketch the stack framework here, going from higher to lower layer:

QTP - Qubit transport protocol (Transport Layer) Responsible for the end to end transmission of qubits.

EMP - Entanglement Management Protocol (Network Layer) Responsible for the generation of entanglement between two nodes that are not directly connected by a link, i.e. not on the same local network.

EGP - Entanglement Generation Protocol (Link Layer) Responsible for the generation of entanglement between to nodes connect by a direct link.

1 Entanglement Generation Protocols

Let us first describe the interface, service, as well as performance criteria of entanglement generation protocols.

1.1 Higher layer to EGP

An EGP supports a single command from the higher layer, namely a request to produce entanglement, which we call a CREATE command. This command includes some desired properties of the entanglement such as for example a minimum fidelity, and a maximum waiting time. In an actual physical implementation, there is a trade-off between these parameters. More time, for example, may allow the underlying implementation to use entanglement distillation to produce higher quality pairs.

CREATE Produce entanglement with a node on the same network (i.e. connected by a link). Arguments supplied are:

Partner ID	ID of the node to generate entanglement with.
Number k	Number of pairs we want to create.
$F_{\min}$	Minimum acceptable fidelity (with high confidence).
$t_{\max}$	Maximum acceptable waiting time before request is completed.
Purpose ID	Identifying the purpose or application at this node (optional, default 0).
Priority	Manual setting of a priority for entanglement production (optional).
Store	Instructs the EGP whether to store the entangled qubit into memory.
create ID	Sequence number identifying this CREATE command.

1.2 EGP to higher layer

Following the reception of the CREATE command, several actions of the EGP are possible. Let us start with the positive outcome, and then consider possible errors. An OK message will be delivered at both nodes involved in the entanglement generation process.

OK Entangled pair has successfully been produced deterministically (heralded). One message per pair created, delivered immediately (best effort) following pair creation. With high confidence, the minimum acceptable fidelity $F_{\min}$ has been met, and the entanglement has been generated within the specified time frame $t_{\max}$. Information about the entanglement generation is provided, including an entanglement identifier. This identifier is required to be globally unique, and agreed upon by A and B. That is, A and B can locally use this entanglement identifier to determine which of their qubits is entangled with the remote node, and also which qubit belongs to which entangled pair. Entanglement identifiers are meant to be shared in the network by higher layer protocols and carry meaning beyond the nodes A and B. An entanglement identifier (Ent_{ID}) consists of:

Origin ID, Partner ID	IDs of the two nodes between which this entanglement is shared.
seqID	Sequence number. Unique (up to wrap around) between A and B, and globally unique when combined with the node IDs.

In addition the OK message also includes the following local information. We remark that Qubit IDs are exclusively local information (akin to the memory address in a computer) and not in general shared between network nodes.

create ID	The create ID that this entangled pair corresponds to.
Goodness	Heuristic estimate for the fidelity of the generated pair.
$t_{Goodness}$	Time when this goodness was established (in EGP, usually the same as generation time).
t_{Create}	Time the pair was produced.
logical_id	Logical ID of the qubit locally holding the entanglement.

Entanglement generation may fail for wide number of reasons, some of which form an immediate error. It may also be that the entanglement later expires, or is discarded of which the EGP will inform the higher layer. Let us start by listing the immediate failure modes, where in all such cases the create ID will be included allowing the higher layer to identify which request has failed.

ERR_UNSUPP Operation not supported. For example, creation of entanglement with the specified minimum fidelity is unattainable, or unattainable within the given time frame, even if the node is not loaded.

ERR_NOTIME Cannot meet the desired fidelity demand within the given time frame due to high load.

ERR_NORES No resources (such as qubits to store the entanglement) available.

ERR_TIMEOUT Failure to produce entanglement within the specified time frame.

ERR_REJECTED Entanglement generation denied by remote node.

ERR_OTHER Failure for unspecified reasons, such as hardware failures.

ERR_CREATE Local error corresponding to a failure to initialize the creation request internally.

In addition, the following failure mode can occur later when an entangled pair is expired. The primary use case of this will be to deal with extremely improbable failures in which recognition of the failure only becomes available after the higher layer has already received an OK message. This allows for a trade-off between speed and certainty in recognizing failure modes. Since entanglement is very short lived, increased certainty can if desired be sacrificed for speed. An example of such a failure would be the case that the outcome information emitted by the heralding station only reaches one of the end nodes and further entanglement attempts are not synchronized.

EXPIRE Expire Qubit ID. Any entanglement associated with Qubit ID has become unavailable.

1.2.1 Questions

- The term "High confidence" is not defined and we need to decide what we mean by that, and also if this is some parameter where/by whom it is determined.
- We need to decide when a node may reject requests to produce entanglement with a neighbour. This likely requires an additional command from the higher layer to set/query a policy.
- It may be advantageous to investigate providing a 'create and measure' functionality. This is to say that the entangled pairs produced under this scheme are immediately measured from the communication qubit rather than moving the state to a storage qubit which could reduce the throughput of entanglement generation and fidelity of the entangled pairs.
- If higher layers are allowed to specify whether entangled qubits should remain in the communication location and there are outstanding generations to satisfy, should the EGP vacate the communication qubit by force to allow for further generation attempts or should this block further generation attempts until higher layers have consumed the qubit?

1.3 Performance metrics

Apart from correctly fulfilling requests, a variety of performance metrics can be considered for EGPs. Not all of these can be simultaneously optimized, but occasionally impose trade-offs. We hereby also draw a distinction between performance metrics of interest to a specific "user" requesting entanglement from the EGP, and the overall performance of the network. Evidently, for all metrics below average, variance, and worst case behaviour is of interest. Once more data is available on how quantum networks are used in practise, one may also consider "typical" values for these metrics.

Let us first consider "user" centric metrics, measuring the experience of one individual user rather than a behaviour of the network as a whole. We remark that nevertheless these metrics are a consequence of the total usage of the network.

Fidelity Quality of the entanglement produced. By design the fidelity has to exceed the minimum requested fidelity $F_{\min}$.

Latency Time between submission of a CREATE request, and an OK response when successful. By design this time may not exceed $t_{\max}$.

In addition, we can consider measures defined by the behaviour of the network when dealing with a large number of requests.

Throughput Number of pairs/s. Refined variants of throughput to be measured include: instantaneous throughput and sustained throughput.

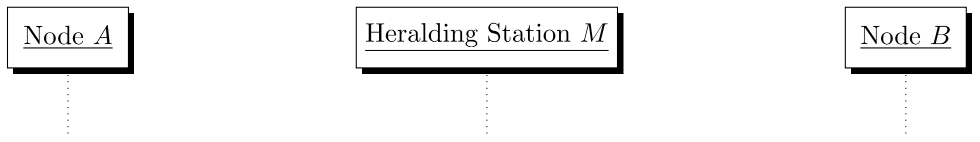
Fairness Difference in performance metrics between requests originating at A and B.

Availability Availability is a concern here if a network node requires resetting and two nodes require re-synchronization at certain time intervals.

We remark that measured values like throughput evidently depend on the request behaviour, including what we will call the *request ratio*, i.e. the number of pairs requested/number of requests total.

2 EGPs based on midpoint heralding protocols (MHPs)

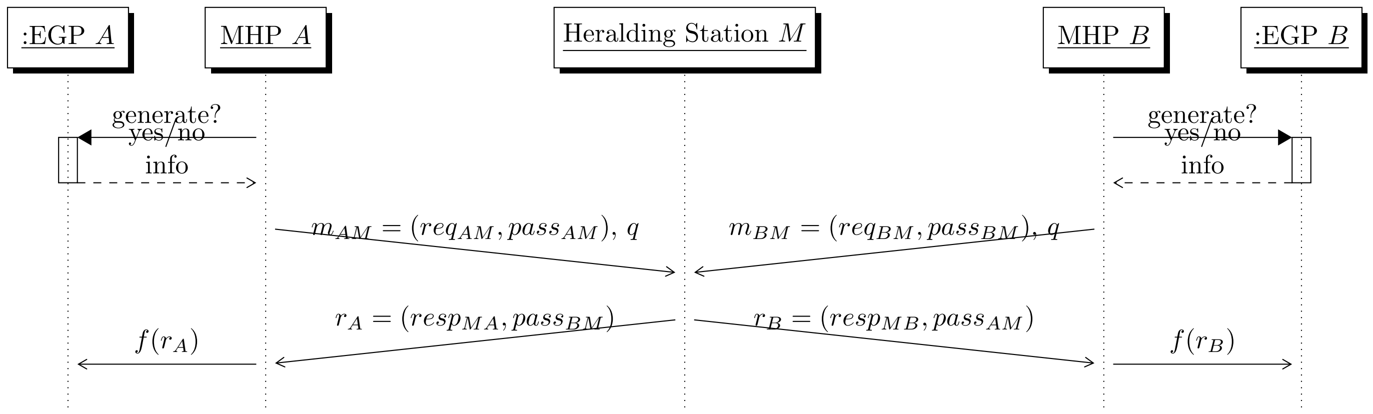
Before proposing a specific EGP, let us first consider a general class of EGPs that are build on top of a system supporting heralded entanglement generation at the physical layer. More precisely, we will consider physical layer protocols that produce entanglement between two nodes A and B, by means of an heralding station M between them, in the following configuration:



Several variations of such schemes exist, such as the single-click, the Barrket-Kok (BK) protocol or even a memory assisted protocol in which entanglement distillation is performed. For simplicity, we will assume a single-click/BK type scheme below, but the following also applies to memory assisted schemes with minor modifications. Evidently, the choice of the physical layer entanglement generation scheme effects the overall performance metrics stated above, as well as the possibility to service requests for entanglement above a specific fidelity $F_{\min}$. For example, certain fidelities may only be attainable by performing entanglement distillation.

Nevertheless, we may cast such physical layer generation schemes in the same following abstract form, upon which our EGPs will be built. We remark that MHPs in our network stack make no high level decisions on when to produce entanglement, scheduling, etc. Also the decision on which MHP to use - eg whether to perform single click, or distill - or even what parameters to set in the single click protocol (such as the bright state population α) are not part of the MHP, but such decisions are left to the link layer EGP, that will use the appropriate MHP (parameters) to obtain the desired service from the physical layer.

Let us thus first give a very abstract description of such protocols, before specifying some assumptions and design alternatives, and stating desired requirements. Here, we have subdivided node into the different units, EGP and MHP.


:EGP A
MHP A
Heralding Station M
MHP B
:EGP B
generate?
yes/no
info
generate?
yes/no
info
$m_{AM} = (req_{AM}, pass_{AM})$, q
$m_{BM} = (req_{BM}, pass_{BM})$, q
$r_A = (resp_{MA}, pass_{BM})$
$r_B = (resp_{MB}, pass_{AM})$
$f(r_A)$
$f(r_B)$

As a simple example, consider the single click protocol with some fixed parameters. Here, m_{AM}, m_{BM} are empty, $resp_{MA}, resp_{MB} \in \{OK, FAIL\}$. Some assumptions and choices were made in the above description:

Assumptions 1. An (essentially) instantaneous association between the classical control messages m below, and the entanglement generation. For this reason, we will write a classical message transmission as simply m, and q for arbitrary quantum signal q, and assume simultaneous arrival. This could be realized by forming a timing association between the classical and quantum signals. To make it clear, how we will use this abstract description later, we will always take $m = (req, pass)$ where req is request information only for the mid point and later protocol specific, and $pass$ is something that will by default always be passed onto the other side (also protocol specific).

2. Midpoint will provide a response $resp$, which includes at least the following information to both A and B: (1) Success or failure of entanglement generation. (2) In case different types of states can be produced, the identity of the state made. (3) A sequence number or equivalent information that allows A and B to determine the ordering of generated pairs.

2.1 Design considerations

Before considering possible protocols, let us abstract some design considerations resulting from the implementation for the non quantum reader:

Basic facts Nodes A and B in the considered implementation are few qubit processors capable of storing and manipulating qubits. Entanglement can be produced probabilistically between one of these qubits, and a travelling photon (q in the above cartoon). When photons from both side arrive simultaneously at the heralding station, a measurement is performed. This measurement will produce 2 possible entangled states with a certain probability p_{succ}, which we will call states 1 and 2 below. This constitutes successful generation. The two types of states can be converted into each other by applying local gates at either A (or B) and thus both states can be considered equally valuable. The measurement at the heralding station can also fail, in which case another attempt must be made. Typical values are $p_{\text{succ}} = 1/2$. The information about success - incl. whether we have state 1 or 2 - or failure is only available at the heralding station, until it is transmitted to A and B.

Not all qubits in the local processors are created equal, in the sense that not all of them can be directly entangled with a traveling photon. We will refer to those that can as communication qubits, and the other as storage qubits. One can transfer the state of a communication qubit to a storage qubit at any time (where we remark that this operation evidently also costs time and introduces noise). In NV in diamond, there is one communication qubits (electron spin) and several storage qubits (called nuclear spins or carbon spins).

Triggering generation Generation of entanglement requires a time synchronized trigger at both A and B that will result in the probabilistic creation of entanglement between the electron spin, and the photon (q in the cartoon above) traveling to the midpoint. If the trigger only occurs at one node, no entanglement can be made. Agreement to produce entanglement at specific time steps thus already has to be realized, requring some communication ahead of time.

Here, we will assume that all low level timing and other synchronization is implemented in the MHP, which is then able to produce entanglement at certain "pre-agreed" i.e. synchronized time instances without additional communication between A and B. As such, the EGP only performs higher level processing. This motivates the choice above that the MHP will poll the EGP, e.g. by reading a specific state variable, on whether entanglement generation is required at a specific synchronized time step. This is in contrast to the EGP sending a signal to the MHP to produce a pair. Since the EGP does not deal with timing synchronization, it cannot know when the actual physical trigger should be produced and hence the MHP could then only save the request until pair production is timely. We remark that this would mean that the MHP would need to keep state of how many outstanding triggers there are, which is not desirable from a design point of view where if $t_{\max}$ has elapsed the EGP may no longer want generation by the MHP. Consequently, we here choose for the MHP to poll the EGP, which does have state on desired generation.

Noise due to generation One may wonder, why entanglement generation is not enabled continuously. That is, attempts to produce entanglement are made all the time, and then the entanglement is either discarded or directly available to the EGP. Two reasons motivated by the physical implementation considered (NV in diamond) make this undesirable: First, triggering an attempt to produce entanglement causes additional noise on the storage qubits. This means that the storage is significantly degraded by trying to produce further entanglement. As such, it is desirable that triggers to attempt generation are only made whenever entanglement is indeed wanted. Second, there are only a small number of storage qubits (presently, 4). If we produce entanglement quickly, by for example triggering an attempt and then immediately transferring the state of the communication qubit to the storage qubit and then proceeding with the next attempt before having heard back from the heralding station, then several storage qubits are needed to support this, making the memory unavailable to other purposes.

For these reasons, MHP will inquire with EGP whether more entanglement is desired, and only then commence production.

Scheduling and flow control The EGP will be responsible for all higher level logic, including scheduling requests. A form of scheduling is flow control which controls the speed of entanglement generation, which hence also falls under higher level logic (see EGP section below).

Memory allocation Decisions on which qubits to use for what purpose lies in the domain of higher level logic, where more information is available on - for example - the number of outstanding requests allowing scheduling decisions including the scheduling of how memory is best used. MHP will hence also not perform memory allocation, i.e., determine which communication qubits or storage qubits to use.

This impacts the types of information included in “info” in the protocol above, which we later take to be of the form (Physical ID Communication Qubit, Physical ID Storage Qubit, Parameters)

Fairness Requests to produce entanglement can originate both at A and B. As such, there can be imbalanced situations in which many more requests originate at A than at B. The EGP should ensure that requests from both nodes have a “reasonable“ chance of being fulfilled.

2.2 Sending classical messages

Above we assumed that there exists a means to transmit classical data between A, B and M. How this is realized is not the objective of this document, and it could be achieved both by a dedicated fiber (possibly using two wavelengths for bidirectional communication), or interspersed with quantum signals on the same fiber.

Here, of interest are merely standard numbers - and the system will need to be implemented to ensure a quality that yields good performance in our link layer protocol. We hence for now consider merely standard numbers:

- Classical channels are bidirectional, meaning data could in principle be sent in both direction at the same time (and, as a relevant consequence, messages can cross and are not ordered in both directions)
- Likelihood of losses: p_{loss} probability of loss (e.g. following standard fiber loss plus electronics if applicable).
- Likelihood of errors: p_{err} probability of error - where we remark that as in other classical communication burst errors are probably dominant.
- Standard delays of interest: propagation delay (over the fiber), transmission delay (incl. delays of the electronics in putting the packets on the fiber), and processing delay, if known. We will assume that given the highly sophisticated electronics and the fact that the rate of classical communication is low due the relatively low repetition rate of entanglement generation attempts, the transmission and classical processing delay are essentially negligible while processing delays due to quantum gates may not be instantaneous.

2.2.1 Enhanced situation

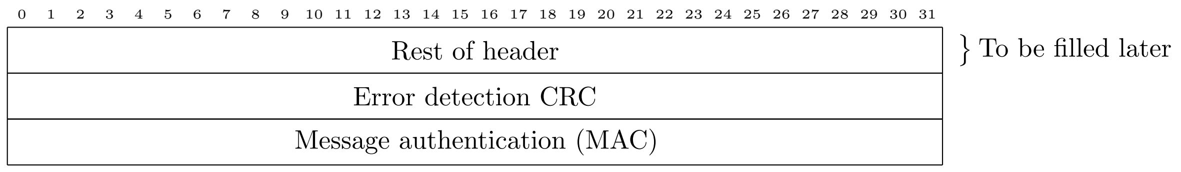
Two standard methods exist to enhance this situation to the following, whose exact form and choice depends on the parameters above. We may also consider running an existing link layer protocol, such as Ethernet over fiber - we do however note that authentication is highly desirable due to control messages leading to local spin photon entanglement generation, and hence otherwise allow unauthenticated messages to manipulate the state of the quantum processing node.

- Error detection: This can be achieved using standard methods, where probably a simple CRC depending on the length of the headers is most appropriate. This will add a number of bits to the messages headers below if employed. For example, for a standard CRC-32 as used in Ethernet, the CRC is computed over the message and stored in a 32 bit header field.

- Message authentication: In this case, this refers to the fact that A knows the messages originate with B (and vice versa). Similarly, M can authenticate messages if desired. Such authentication can be realized using a message authentication code (MAC) (see eg **?**). These can be realize with varying levels of security. If A and B share consumable key (such as for example generated by QKD), they can afford to use one-time MAC which - similar to a one-time pad - offers information-theoretic security. Such MACS, for example based on two-universal hashing, can in principle be fast (see e.g. **?** needing however various levels of key), although it is a question whether they are fast enough to be useful in this context. We remark that also weaker forms of authentication could be acceptable as they merely form a means of protection and would need to be broken in a very short time frame to have a practical impact.

2.2.2 Enhanced situation

Based on the general shape of such protocols above, one can now consider a slight "enhancement" of a protocol of this form - like single-click - that makes explicit some (probably obvious) failure modes, and produces a total ordering of pairs that A and B agree upon, even if some messages may go missing.



3 Candidate EGPs

We will now consider three classes of link layer EGPs build on top of MHPs. Strictly speaking, we will consider two non trivial implementations, and one rather ad-hoc one with rather trivial features for comparison.

The present objective is to implement these in simulation to:

1. Assess their relative performance with respect to each other. Performance metrics have been specified above.

2. Assess the effect of specific design choices made in each of these protocols.

3. Study the relevance of parameter demands, specifically also the required quality of the classical communication and its timing so it can be implemented.

The main difference between the two more sophisticated EGPs is where queuing and scheduling is done, demanding more or less decision power at the heralding station.

3.1 Node Centric: Distributed Queue

In the first scenario, the heralding station is essentially passive and performs no higher level functions. Before sketching the protocol, let us first outline its ingredients next to the MHP. In addition to storing the parameters supplied with a CREATE request, the protocol will keep the following information about each request:

- origin_id: ID of the node where the CREATE request was made
- create_time: (local) time at which this request was received
- ready_flag: marks whether this request is ready to be executed
- min_time: minimum time at which the request may be executed

To ensure fairness, the protocol will also use the parameters win_A, f_A and win_B, f_B at A and B, which we will denote as the window size win and fairness f respectively.

3.1.1 Priority Queues (PQ)

In principle, we may want to give priorities to certain requests. This will be accomplished by adding them to different types of queues $\mathcal{Q} = \{Q_1, \ldots, Q_L\}$. Each queue can contain a maximum of x items simultaneously, where x is a parameter. Each CREATE requests is assigned a queue number by the scheduler (see below), and receives a queue id (j, id_j) where j indicates the designated queue Q_j and id_j is a unique queue id. When the queue Q_j is clear from context, we will refer to id_j as the "queue id" of the request. For the totality of queues $\mathcal{Q}$, we will refer to (j, id_j) as the *absolute queue id*, and use $(j, id_j) \in \mathcal{Q}$ to indicate a request with that absolute queue id is in the queue. The queue id is required to obey the following properties:

- (Total order) Items on each queue are obey a total order determined by id_j.
- (Arrival time) Let t_1 and t_2 denote the create_time of requests 1 and 2 respectively. Let id_1 and id_2 their queue id's respectively. If both requests are added to the same queue Q_j, and $t_1 < t_2$, then $(id_2 - id_1) \mod x > 0$. That is, if a CREATE request arrives earlier, it will also receive a lower queue id.

For simplicity, we will for now assume there is only one queue, i.e., $L = 1$.

3.1.2 Distributed Queue Protocol (DQP)

The objective of the DQP is to obtain shared - i.e. agreed upon queues - at both nodes. That is, both A and B hold local queues $\mathcal{Q}^A = \{Q_1^A, \ldots, Q_L^A\}$, and $\mathcal{Q}^B = \{Q_1^B, \ldots, Q_L^B\}$ respectively, which are synchronized using the DQP. Additions can be made by either A or B invoking the DQP to $(id_j, ok) = ADD(j, creq)$ add a specific CREATE request $creq$ to queue Q_j. "ok" indicated success or failure: failure can occur if (1) no acknowledgements are received within a certain time frame, i.e. a timeout occurs (2) the remote node rejects addition to the queue. Success means that the request to create entanglement is placed into $\mathcal{Q}^A$ and $\mathcal{Q}^B$ such that the following are satisfied:

- (Equal queue number) If a request is added by A as $(j, id_j) \in \mathcal{Q}^A$, then it will (eventually) be added at B with the same absolute queue id $(j, id_j) \in \mathcal{Q}^B$ (and vice versa).
- (Unique queue id) If a request is placed into the queue by either A (or B), then it is assigned a unique queue number. That is, if $(j, a) \in \mathcal{Q}^A$ and $(j, a') \in \mathcal{Q}^A$ reference two distinct CREATE requests, then $a \neq a'$.
- (Consistency) If $(j, id_j) \in \mathcal{Q}^A$ and $(j, id_j) \in \mathcal{Q}^B$ then both refer to the same request. This is implied by the previous two conditions, but added for clarity.

- (Fairness) If A (or B) is issuing requests continuously, then also the other node B (or A) will get to add items to the queue after A (or B) in a "fair manner" as determined by the window size win_A (win_B) and fairness f_A (f_B). More precisely, if $a_1, \ldots, a_N$ are CREATE requests submitted at A, and $b_1, \ldots, b_L$ are CREATE requests submitted at B - all assigned to the same queue Q_j but not yet added - , then the final ordering of the requests on the queue obeys $a_1, \ldots, a_m, b_1, \ldots, b_k, a_{m+1}, \ldots$ with $m \leq win_A$ and $f_B \leq k \leq win_B$ (and vice versa, using f_A). (NOTE: f_A, f_B not yet implemented in simulation)
- (Scheduling information) A request on the queue originating at A (B) is marked ready when ack from B (A) has been received.

Recall that each request receives a minimum to be executed time min_time, which we will choose to be the expected propagation delay between A and B, decreasing the likelihood A or B wants to produce before the other node is ready as well. No penalty other than reduced performance due to increased decoherence results if one of them goes too early.

Given we have only two nodes, the above can easily be realized by one node being the master controller of the queue marshalling access to the queue, and the other the slave controller. Extensions to multiple nodes are more complex, and indeed a motivation to consider heralding station centric protocols in this case.

3.1.3 Scheduler

The scheduler fulfills the following arbitrage functions, where we remark that for CREATE requests that demand multiple EPR pairs, only one request is added to the queue, and hence NEXT (see below) will return multiple pairs to be produced for the same request when called successively.

- GET QUEUE(*creq*): Once a request has been submitted, GET QUEUE deterministically determines which queue Q_j to assign the CREATE request *creq* to. This may depend on the details of the request, such as for example $t_{\max}$, or $F_{\min}$.
- NEXT: Selects the next request from the local set of queues $\mathcal{Q}$ to serve, if any. Specifically, NEXT will determine:
  - Flag: True when a request is ready to be served.
  - Absolute queue id (and corresponding request details) of request to be served.
  - Parameters to use in the MHP depending on the number of type of outstanding requests.
  - In cooperation with QMM, determine communication and storage qubits.

  The scheduler will also marshal a form of flow control, that is we will speed up or slow down the production of entanglement. This will be implemented in the simulation for now as a stub, for first testing to be filled in later. Speeding up or slowing down is relevant if the local quantum memory is full, i.e., we can no longer produce entanglement since insufficiently many free qubits are available. To do this we remark that by a standard Hoeffding bounds, A (or B) can likely produce entanglement except with probability ε, if the number of qubits A (or B) has free e_A (or e_B) satisfies

$$n \leq \frac{e_A}{p_{\text{succ}} + \sqrt{\frac{1}{2n} \ln\left(\frac{1}{\varepsilon}\right)}}, \tag{1}$$

  where p_{succ} is the probability of successful entanglement generation and n is the number of qubits currently in flight (i.e. we do not yet know the outcome of entanglement generation). Below we will say that A (or analogously B) can likely produce entanglement, if the above expression is satisfied.

3.1.4 Quantum Memory Management (QMM)

While not part of the actual protocol, the QMM can be asked to (smartly) allocated a fresh qubit and to convert logical qubit IDs to physical Qubit IDs and vice versa.

3.1.5 Fidelity estimation unit (FEU)

We are planning to perform error detection by including later an online fidelity estimation protocol, that together with already available information from the experiment allows us to make a guess for the fidelity communicated to higher layers via the Goodness in the Entanglement ID.

May intersperse CREATE requests of its own to update its online fidelity estimate of the EGP.

3.1.6 Protocol Sketch

Let us now describe the local protocol running at each node, once a CREATE request is received. For clarity we will describe it for node A, but it applies to both analogously:

Protocol 1 EGP - Node A (B analogous exchanging A and B)

Inputs. CREATE request: $creq = (\text{Node ID } B, n, F_{\min}, t_{\max}, \text{Purpose ID}, \text{Priority}, \text{CREATE SEQ ID})$

Goal. Produce entanglement, or otherwise declare failure.

Initialization. Initialize outstanding generation list G. *The protocol:*

1. **Adding to Queue:**
   (a) Ask scheduler which queue this request should be added to: $Q_j = \text{GET QUEUE}(creq)$
   (b) Try to add request to the queue using the DQP: $(id_j, ok) = ADD(j, creq)$.
   (c) If $ok = timeout$ error, issue ERR_NOTIME and stop.
   (d) If $ok = reject$ error, issue ERR_REJECTED and stop.
   (e) Otherwise the request has been added to the Distributed Queue.
2. **Trigger pair (polled by MHP_NC):**
   (a) Ask the scheduler whether entanglement should be made and for which request: $(flag, (j, id_j) \equiv req, param, comm\_q, storage\_q) = NEXT$. For this end, the scheduler will employ its priority policy, as well as perform flow control depending on whether B is likely to produce entanglement.
   (b) If there is a generation waiting to be satisfied:
      i. Add (j, id_j) to outstanding generation list G.
      ii. Construct response for MHP_NC $(flag, (j, id_j), comm\_q, storage\_q, param)$.
      iii. Provide the response to the MHP_NC.
   (c) Otherwise if there are no generations to perform provide $(False, None, None, None, None)$.
3. **Handle reply (message from MHP_NC):**
   (a) Retrieve message from MHP including: result of generation $r \in \{0, 1, 2\}$, MHP_seq, absolute queue id (j, id_j), and protocol error flag $proto_{err}$.
   (b) If the specified absolute queue ID is not found locally then the request may have timed out or expired:
      i. Free the reserved communication qubit in the QMM and stop handling.
   (c) Otherwise there is an absolute queue ID included in the response that is associated with an active request:
      i. If $proto_{err}$ is not OK, a non quantum error occurred and no entanglement was produced. Stop handling reply. (NOTE: we may wish to deal with such errors differently?)
      ii. If $r = 0$ we failed to produce entanglement, stop handling reply.
      iii. If the MHP_seq is larger than the expected sequence number, (partially) ERR_EXPIRE the outstanding requests on G. Remove them from G. Stop handling reply.
      iv. If MHP_seq is smaller than the expected sequence number, ignore reply. Stop handling reply.

Protocol 1 (cont.) EGP - Node A (B analogous exchanging A and B)

v. Update next expected MHP_seq sequence number (increment current one modulo x).

vi. Remove (j, id_j) from outstanding generations list G.

vii. We made a pair! If $r = 2$ and we are the origin_ID node of this request, apply correction information to transform state 2 to state 1, if we are the peer then we instruct the scheduler to suspend subsequent generation attempts until we believe the request originator has completed correction.

viii. Look up queue item (j, id_j):

A. If create_time + $t_{\max}$ > current time or the request is not stored locally anymore, issue ERR_TIMEOUT and remove item from queue. Stop handling reply.

B. Get fidelity estimate F_{est} from FEU.

C. Issue OK with Entanglement ID: (A, B, MHP_seq), $logical_id$, Goodness F_{est}, and $t_{\text{Goodness}} = t_{\text{Create}} = now$.

D. If $k = 1$, i.e. this was the last pair to be produced for this request, remove item from queue.

E. If $k > 2$, decrement k on queue item.

Let us now describe how we will use the MHP, recall that it will be automatically executed at specific time steps and poll the EGP for the entanglement generation flag (or more accurately list of outstanding "flags"). Here, we will assume that in addition to the flag y/n whether to produce entanglement, we will also supply the MHP with (1) the physical ID of the communication qubit (2) the physical ID of the storage qubit (3) any parameters such as the bright state population for entanglement generation (4) the absolute queue ID.

We specify the MHP by filling in the default messages given above. Again we will for simplicity do this only for A but the same holds for B.

Protocol 2 MHP_NC - MHP for use with the Node Centric EGP above

Goal. Create entanglement using a mid point heralding protocol.

Initialization. Initialize sequence numbers. Start timer.

The protocol - executed at each time step:

1. **Node A:**
   (a) Poll EGP to obtain: $flag, comm\_q, storage\_q,$ absolute queue id$aid, param$
   (b) If $flag$ is true, i.e., we want to make entanglement:
      i. Reserve $comm\_q$ and $storage\_q$.
      ii. Initialize $comm_q$, and produce spin-photon entanglement. Store spin state in $storage\_q$. If any failures occur, send $mhp_{err} = GEN\_FAIL$ back to the EGP and skip to next time step.
      iii. Use $req_{AM} = (Produce)$ and $pass_{AM} = (aid)$.
   (c) If $resp_{MA}$ returns from midpoint: Choose f to pass the following information up to the EGP: r, MHP seq and aid.
2. **Heralding station:**
   (a) If messages from A and B do not arrive within the same time interval, send $resp_{MA} = resp_{MB}$ including $mhp_{err} = TIME\_MISMATCH$. Skip to next time window.
   (b) If the aid from A does not match the aid from B, send $resp_{MA} = resp_{MB}$ including $mhp_{err} = QUEUE\_MISMATCH$
   (c) "Otherwise" Execute quantum swap.
   (d) Use $resp_{MA} = (r, MHPseq, aid, info)$ with $r \in \{0, 1, 2\}$, where 0 denotes failure and 1 and 2 denote the creation of states one and two respectively. If if $r \in \{1, 2\}$ a unique and increasing sequence number MHP seq is chosen by the heralding station and send to both A and B. The heralding station may send additional information (eg the moment of detection) to allow a more accurate fidelity guess.

3.2 Heralding Station Centric: Arbitrage by heralding station

3.3 Happy go lucky