Link Scheduling Notes

mdskrzypczyk

August 2019

1 Device notes

- Matteo says that any NV device can be brought to the T_1/T_2 parameters used in the link layer simulations through careful tuning.
  - Might mean it's worth focusing on a homogenous network of nodes that decohere at the same rate. This means that only the initial fidelity matters depending on the distance of the links.
  - SWAP'd links will then degrade at the same rate as the individual ones between nodes, can apply an exponential fitting to track the loss of fidelity.
- Also said that the parameters may fluctuate between 30% and 100% of those used in the simulations, so can also investigate heterogenous networks.
- Consider storing the links with better fidelity first in the carbon due to the great storage time. This will *probably* maximize the resulting swapped fidelity.

1.1 Decoherence for single/double electron/carbon storage

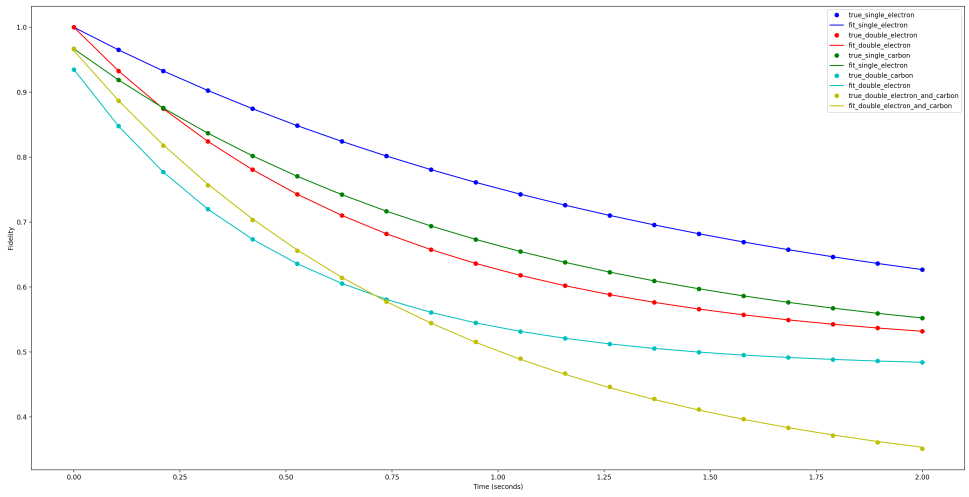

Figure 1: Plot of decoherence on an entangled pair of qubits for cases where a single qubit is in the electron/carbon and the other is stored perfectly, both are in the electron, both are in the carbon, and one in electron while other in carbon. All bell pairs initiated at $F = 1$ with move to storage performed immediately after generating.

This plot was generated using the following values for T_1/T_2 times sourced from the experimental realizations discussed in *A Link Layer Protocol for Quantum Networks*.

From the collected data it also appears that if we fit the curves with $f = ae^{-bt} + c$ for storage in electron/electron, carbon/carbon, and electron/carbon, we can compute the single electron/carbon

	Electron T_1	Electron T_2	Carbon T_1	Carbon T_2
Duration/Time	1h	1.46s	10h (inf)	1s

decoherence using the same a, c but computing $b_{single} = \frac{1}{2} b_{double}$ and obtaining b_{double} for electron/carbon by simply summing b_{single} for both the electron and carbon decoherence.

Additional plots of decoherence consider inserting a delay before the SWAP occurs from the electron to the carbon.

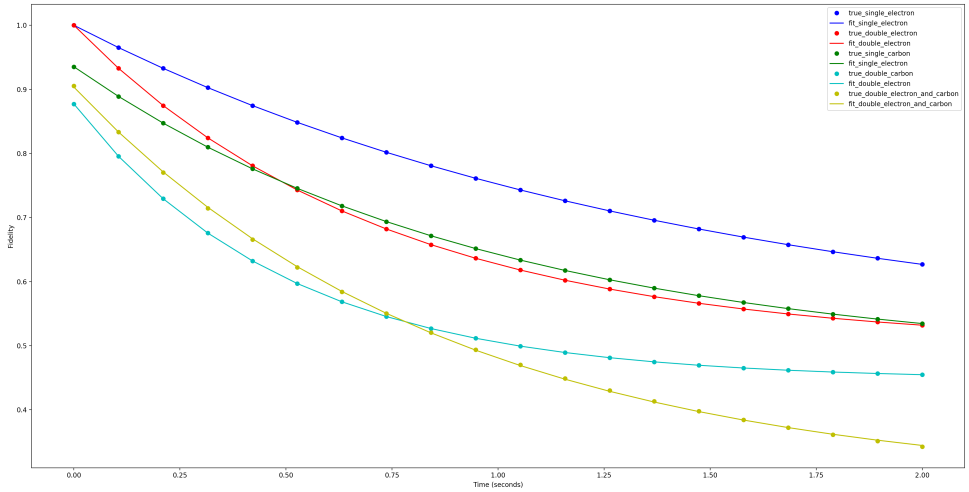

Figure 2: Plot of decoherences with SWAP delay of $100ms$.

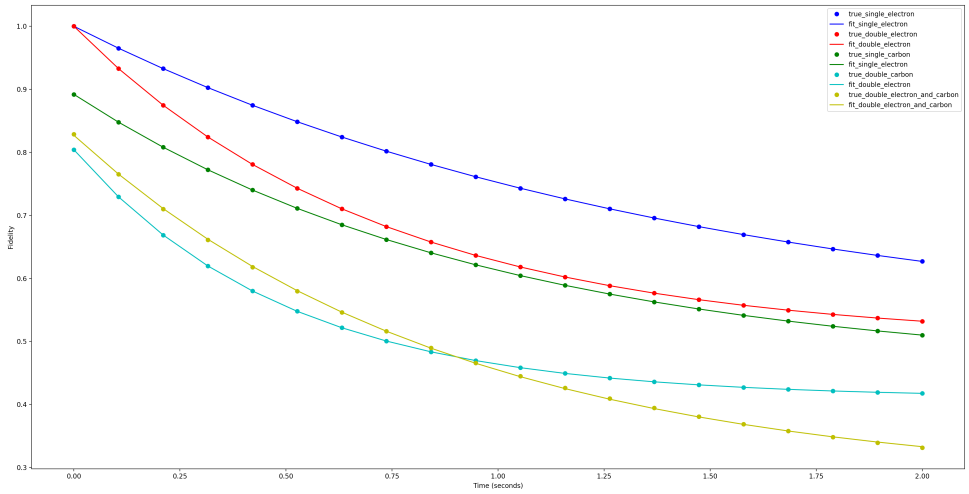

Figure 3: Plot of decoherences with SWAP delay of $250ms$.

2 Cost functions

Objective functions to optimize with choice of link scheduling.

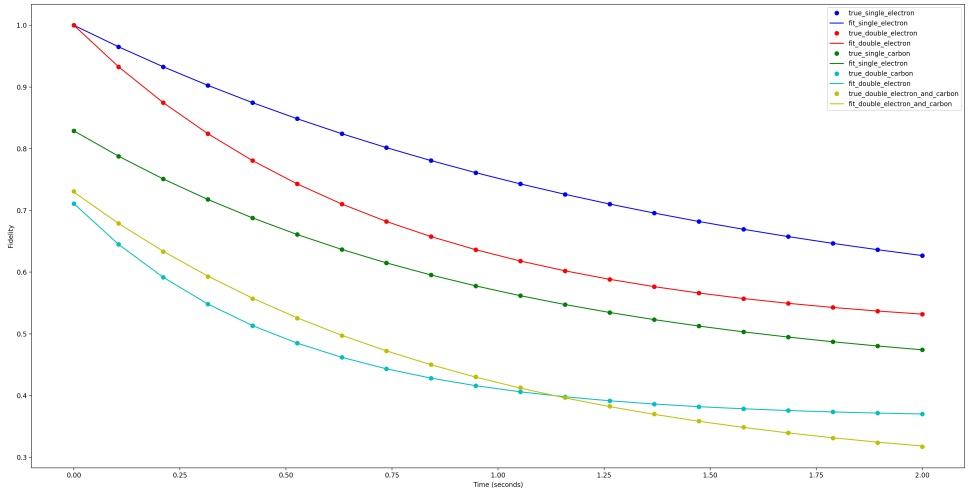

Figure 4: Plot of decoherences with SWAP delay of $500ms$.

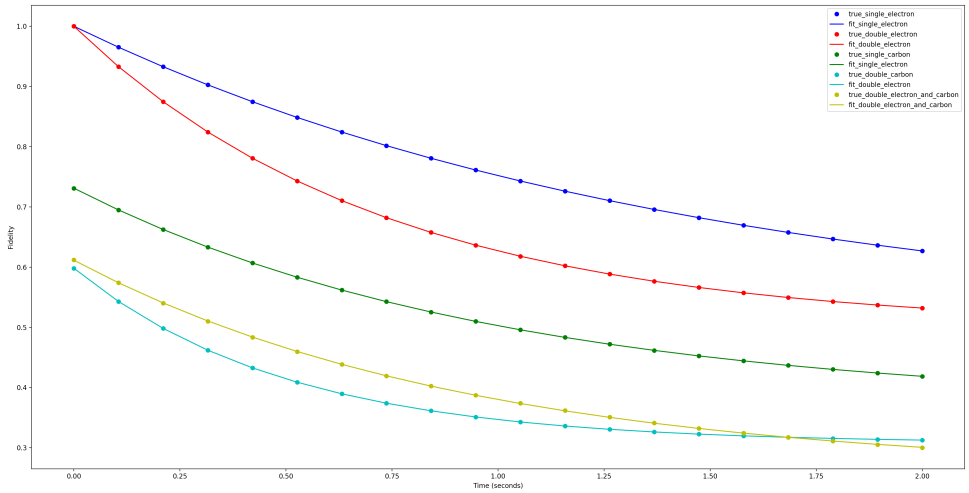

Figure 5: Plot of decoherences with SWAP delay of $1000ms$.

2.1 Minimized Weighted Idle Time

$$\min d = \sum_{i=1}^{n-1} w_{i+1}|e_i - e_{i+1}| + max_i(e_i)(w_1 + w_{n+1}) - w_1 e_1 - w_{n+1} e_n$$

Where:

- n is the number of edges in the chain
- e_i is the time at which link i is expected to be generated
- w_i is a decoherence weight associated with node i
- $\sum$ term represents the weighted decoherence introduced by the nodes internal to the chain (2:n)
- $max_i(e_i)$ is the completion time of the last link

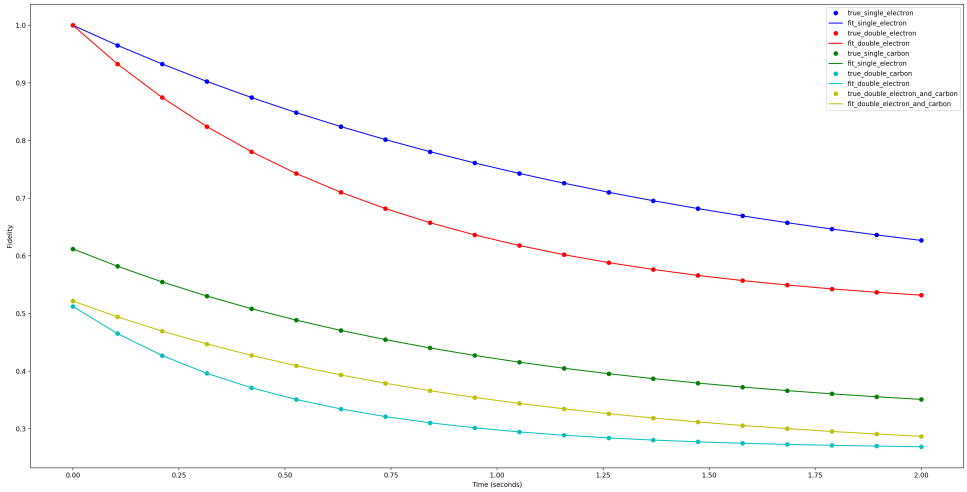

Figure 6: Plot of decoherences with SWAP delay of $2000ms$.

- Final terms represent the decoherence introduced on the qubits held by the end nodes as they must wait until the full link has been established to begin using

Observations:

- Preferable to have end links generated last as generating them earlier accumulates more decoherence (dependent on weights)

2.2 Minimized Total Fidelity Loss

$$\min F_{loss} = \sum_{i=1}^{n} F_{0,i} e^{-w_i t_i}$$

Where:

- F_{loss} is the total lost fidelity
- $F_{0,i}$ is the initial fidelity of a generated link
- w_i is the rate of decoherence of a link
- t_i is the amount of time the link decoheres for

Problems:

- Metric does not consider how the links decohere after they have been SWAPPED, w_i is a property of the originally generated link, not the nodes holding the qubits

3 Greedy Heuristics

- Place links that take less time to generate ahead of current link and links that take less time behind the current link.
  - Can this actually put us in a position where decoherence puts fidelity of a link too low to be useful?
- Place links that have a higher fidelity by the time both links should be done earlier in the schedule

4 Coordination

4.1 Timeslot Scheduling

Share schedules of nodes along path and insert slots for when time should be spent generating a link.

4.1.1 Why Timeslot Scheduling?

- We want a coordination mechanism that can take into account a node's availability to establish entanglement in the network.
- This may include applications that are running at the node which cannot be interrupted or other links the node is helping establish.
- We also want to reduce additional communication for coordinating link building, ie. avoid communication between nodes of the form "I am done with link X are you available to start link Y?" as this might go on indefinitely.
- Timeslot scheduling allows us a simple mechanism to stop generating a link. If we have not generated a shorter segment by the end of the slot, we give up and can instruct other nodes to give up as well.
- Timeslot scheduling can be done in a centralized manner where an external central authority knows all the schedules of the nodes or requests them or it can be done in a distributed manner where the nodes along the path share their schedules with one another.

4.1.2 Execution Techniques

- Get individual machine schedules.
  - Schedule cannot be represented infinitely into the future (messages are finite in size, base size on the max time of generation request?
  - Need an efficient representation that permits taking intersections for creating link schedules
  - Prune slots that are occupied or too short for generating a given link. "Too short" can have a few meanings, below average or below some fraction of average time to generate. (Check probability distribution, geometric?)
  - Represent schedule for each communication qubit on the link
- Take an intersection to form availability schedules for each link.
  - How can we take into account several communication qubits for this part?
  - Reserving a slot in a link schedule affects the adjacent link schedule availability.
- Determine the order of the links and which slots should be used.
  - Prune to isolate valid pairs of adjacent slots that allow SWAP-ing to achieve reasonable fidelity.
    - "Window" adjacent schedules to find pairs of slots. Window size should tolerate occupied slots between available slots. May be based on achievable fidelity and how low we can accept the fidelity to become before the second link generates.
    - Can we order these slot pairs by a greedy heuristic?
    - Can also order based on how much faster/slower a link would degrade the sooner we can push a SWAP through to a device that decoheres less.
  - There may be several "starting" points within a link availability schedule, how to pick which one?
    - Consider the timespan of generating all the links?
    - Construct a naive order assuming full availability and then slot?
    - Construct the order based on the available slots? Start with the earliest slot? Start with the worst link?
    - Brute force is $O(l2^n)$ if an end node's schedule has l starts and we use the closest two slots (before/after) for n nodes

    - Greedy has $O(nl)$ if an end node has l different starts and we always select the next one simply
  - In order to calculate fidelities, can require devices to provide some sample points from a Fidelity curve of the qubits. Using a granularity at least as fine as that used for the timeslots can allow us to determine what the fidelity of an entangled pair would be by the time we expect the next one to be ready. The problem with this is that it does not provide us information about how the SWAP'd links behave. Would be unreasonable to request these points for every possible SWAP'd link. Alternatively, require nodes to supply (a, b, c, t_0) for the fit $F = ae^{-bt} + c$ and the initial fidelity of the generated link $F_0 = ae^{-bt_0} + c$. This also does not seem to imply an immediate way to find out how the SWAP'd link degrades (actually from a short simulation with two nodes that have the same parameters it seems like the weight in the exponent just doubles when stored on two devices, if all nodes are homogenous then the same curve can be used but we need to calculate the new SWAP'd fidelity and resume decoherence from the corresponding point in the curve).

- How to handle requests for several pairs?
  - Instruct nodes to have a decided link scheduling recurring to meet the max time requirement. "Do schedule X every 10 slots".
  - Using multiple routes? This can incur a significant communication overhead if a request is cancelled.

5 Questions

6 Does the order of SWAP matter?

Three links with fidelities F_1, F_2, F_3. Equation for SWAP'd fidelity:

$$F_{new} = F_1F_2 + \frac{(1-F_1)(1-F_2)}{3}$$

Let $F' = F_1F_2 + \frac{(1-F_1)(1-F_2)}{3}$ be the SWAP'd fidelity of links 1 and 2. Then:

$$F'F_3 + \frac{(1-F')(1-F_3)}{3}$$

$$= (F_1F_2 + \frac{(1-F_1)(1-F_2)}{3})F_3 + \frac{(1-(F_1F_2 + \frac{(1-F_1)(1-F_2)}{3}))(1-F_3)}{3}$$

$$= F_1F_2F_3 + \frac{F_3(1-F_1)(1-F_2)}{3} + \frac{(1-F_1F_2 - \frac{(1-F_1)(1-F_2)}{3})(1-F_3)}{3}$$

$$= F_1F_2F_3 + \frac{(F_3 - F_1F_3)(1-F_2)}{3} + \frac{(1-F_3) - F_1F_2(1-F_3) - \frac{(1-F_3)(1-F_1)(1-F_2)}{3}}{3}$$

$$= F_1F_2F_3 + \frac{F_3 - F_2F_3 - F_1F_3 + F_1F_2F_3}{3} + \frac{1 - F_3 - F_1F_2 + F_1F_2F_3 - \frac{(1-F_3-F_1+F_1F_3)(1-F_2)}{3}}{3}$$

$$= F_1F_2F_3 + \frac{1}{3}(1 - F_1F_2 - F_2F_3 - F_1F_3 + 2F_1F_2F_3) - \frac{1}{9}(1 - F_1 - F_2 - F_3 + F_1F_2 + F_2F_3 + F_1F_3 - F_1F_2F_3)$$

$$= \frac{2}{9} + \frac{1}{9}(F_1 + F_2 + F_3) - \frac{4}{9}(F_1F_2 + F_2F_3 + F_1F_3) + \frac{16}{9}F_1F_2F_3$$

Substituting fidelities of 1 in provides what we expect:

$$\frac{2}{9} + \frac{1}{9}(3) - \frac{4}{9}(3) + \frac{16}{9}$$
$$= \frac{2}{9} + \frac{3}{9} - \frac{12}{9} + \frac{16}{9}$$
$$= \frac{5}{9} + \frac{4}{9}$$
$$= 1$$

In short, the order of SWAP does *not* matter. What may matter is if SWAP-ing and holding that link between the two nodes pushes it below a permitted Fidelity due to the rate of decoherence.