Supplementary Material for "Experimental demonstration of entanglement delivery using a quantum network stack"

M. Pompili[1,2], C. Delle Donne[1,2], I. te Raa[1], B. van der Vecht[1], M. Skrzypczyk[1], G. Ferreira[1], L. de Kluijver[1], A. J. Stolk[1], S. L. N. Hermans[1], P. Pawełczak[1], W. Kozlowski[1], R. Hanson[1,3], and S. Wehner[1,4]

[1]QuTech & Kavli Institute of Nanoscience
Delft University of Technology, 2628 CJ Delft, The Netherlands

[2]These authors contributed equally to this work

[3]Correspondence to: R.Hanson@tudelft.nl

[4]Correspondence to: S.D.C.Wehner@tudelft.nl

1 Supplementary note 1: Pre-computed TDMA schedule

TDMA network schedules are redundant in a one-link network. Time-binning network activity also forces nodes to only process entanglement requests at the beginning of a time division, thus introducing latency and idle time. Particularly, longer time bins potentially result in entanglement requests to wait longer to be processed. However, an application asking for multiple entangled pairs with just one request would experience smaller average latencies, as all pairs—but the first one—would be generated in close succession.

In our experiments, TDMA schedules are just a constant division of 20 ms time bins, each of which is reserved to the only application running. We chose the duration of the time-bin—somewhat arbitrarily, given the small effect on our experiments—to be equal to 1000 communication cycles between the device controller and the network controller ($20\,\text{ms} = 1000 \times 20\,\mu\text{s}$).

2 Supplementary note 2: Single qubit gates implemented

At the physical layer, we implement real-time rotations around the X and Y axes of the qubit Bloch sphere, using a resolution of $\pi/16 = 11.25°$. That is, the link layer can request any rotation that is a multiple of $\pi/16$ around either the X or Y axis. The different rotations are performed using Hermite-shaped pulses (as described in Ref. [1]) of calibrated amplitude. The choice of X(Y) rotation axis is implemented using the I(Q) channel of the microwave vector source.

While supported at the link layer, our physical layer currently does not implement Z axis rotations. Such rotations around the Z axis could be implemented by virtual rotations of the Bloch sphere: a π pulse around the Z axis is equivalent to multiplying future I and Q voltages by -1. By keeping track of the accumulated Z rotations, and by adjusting I and Q mixing accordingly, one can perform effective Z rotations with very high resolution and virtually no infidelity. The AWGs currently in use have the required capabilities, and the implementation of said Z gates is planned for the near future.

3 Supplementary note 3: Clock sharing and AWG triggering over longer distances

One of the technical challenges of realizing a large scale quantum network is synchronizing equipment at the physical layer across nodes. The synchronization is required to generate entanglement—the photons from the two nodes need to arrive at the same time at the heralding station (compared to their duration, 12 ns for NV centers in bulk diamond samples); failing to do so would reduce (or even remove) their indistinguishability, which is required to establish long-distance entanglement [1]. Our two nodes are located in a single laboratory, on the same optical table, approximately 2 m apart. This allows for some simplifications, for the purpose of demonstrating entanglement delivery using a network stack, which would not be possible over longer distances. Specifically:

1. We use a single laser—the client's—to excite both nodes, as in Ref. [1]. Over longer distances, one would need to phase-lock the excitation lasers at the two nodes to ensure phase-stability of the entangled states.

2. The Device Controllers (ADwin Pro II microcontroller) are triggered every 1 µs by the same signal generator, advancing the state machine algorithm that implements the physical layer. This ensures that the two microcontrollers have a common shared clock. Over longer distances, one could use existing protocols (and commercially-available hardware) to obtain a shared clock [2], and use that to trigger the microcontrollers.

3. The two AWGs need to be triggered to play entanglement attempts. In our implementation, one device controller—the server's—triggers both AWGs. This ensures that the triggering delay between the two AWGs is constant, and we can therefore calibrate it out. Triggering the AWGs with two independent microcontrollers would result in jitter (realistically on the order of nanoseconds). Over larger distances, one could derive—from the shared clock—a periodic trigger signal that is gated by the microcontroller at each node. In this way the microcontroller can decide whether the AWG will be triggered on the next cycle, but the accuracy of the trigger's timing will be derived from the shared clock between the nodes, rather than from the microprocessor.

4. The phase stabilization scheme we use, developed in Ref. [1], is designed to work at a single optical frequency (in our case, the 637 nm emission frequency of the NV center). Over longer distances, conversion of the NV center photons to the telecom band will be necessary to overcome photon loss. The phase stabilization scheme will therefore need to be adapted to new optical frequencies used.

For reference, our client (server) is based on node Charlie (Bob) of the multi-node quantum network presented in Ref. [1].

4 Supplementary note 4: Applications

Following are pseudocode sequences that describe the applications executed via the quantum network stack. Their purpose is to outline how the applications were executed (sweep order), and the differences between the three applications.

Listing 1: Full quantum state tomography.

```
# The common list of correlators that are going to be measured (client, server).
correlator_list = [
    (-X, -X), (-X, -Y), (-X, -Z), (-X, +X), (-X, +Y), (-X, +Z),
    (-Y, -X), (-Y, -Y), (-Y, -Z), (-Y, +X), (-Y, +Y), (-Y, +Z),
    (-Z, -X), (-Z, -Y), (-Z, -Z), (-Z, +X), (-Z, +Y), (-Z, +Z),
    (+X, -X), (+X, -Y), (+X, -Z), (+X, +X), (+X, +Y), (+X, +Z),
    (+Y, -X), (+Y, -Y), (+Y, -Z), (+Y, +X), (+Y, +Y), (+Y, +Z),
    (+Z, -X), (+Z, -Y), (+Z, -Z), (+Z, +X), (+Z, +Y), (+Z, +Z)]

def client_application():
    for rep in range(125):
        for corr in correlator_list:
            client_basis = corr[0]
            # Establish the entangled state.
            client_qubit = create_ent(
                with=Server,
                req_type=Keep,
                min_fidelity=0.8)
            # Perform the required rotation.
            client_qubit.rotate_basis(client_basis)
            # Measure the qubit, store the result.
            outcomes[rep, corr] = client_qubit.measure()

def server_application():
    for rep in range(125):
        for corr in correlator_list:
            server_basis = corr[1]
            # Establish the entangled state.
            server_qubit = receive_ent(
                with=Client,
                req_type=Keep,
                min_fidelity=0.8)
            # Perform the required rotation.
            server_qubit.rotate_basis(server_basis)
            # Measure the qubit, store the result.
            outcomes[rep, corr] = server_qubit.measure()
```

Listing 2: Delivery of entangled states at varying fidelity and rate.

```
# The common list of correlators that are going to be measured (client, server).
correlator_list = [
    (-X, -X), (-X, +X), (-Y, -Y), (-Y, +Y), (-Z, -Z), (-Z, +Z),
    (+X, -X), (+X, +X), (+Y, -Y), (+Y, +Y), (+Z, -Z), (+Z, +Z)]

# The target fidelities to generate.
fidelity_list = [0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80]

def client_application():
    for rep in range(125):
      for fid in fidelity_list:
        for corr in correlator_list:
          client_basis = corr[0]
          # Establish the entangled state.
          client_qubit = create_ent(
                with=Server,
                req_type=Keep,
                min_fidelity=fid)
          # Perform the required rotation.
          client_qubit.rotate_basis(client_basis)
          # Measure the qubit, store the result.
          outcomes[rep, fid, corr] = client_qubit.measure()

def server_application():
    for rep in range(125):
      for fid in fidelity_list:
        for corr in correlator_list:
          server_basis = corr[1]
          # Establish the entangled state.
          server_qubit = receive_ent(
                with=Client,
                req_type=Keep,
                min_fidelity=fid)
          # Perform the required rotation.
          server_qubit.rotate_basis(server_basis)
          # Measure the qubit, store the result.
          outcomes[rep, fid, corr] = server_qubit.measure()
```

Listing 3: Remote preparation of a qubit on the server.

```
# The common list of correlators that are going to be measured (client, server).
correlator_list = [
    (-X, -X), (-X, -Y), (-X, -Z), (-X, +X), (-X, +Y), (-X, +Z),
    (-Y, -X), (-Y, -Y), (-Y, -Z), (-Y, +X), (-Y, +Y), (-Y, +Z),
    (-Z, -X), (-Z, -Y), (-Z, -Z), (-Z, +X), (-Z, +Y), (-Z, +Z),
    (+X, -X), (+X, -Y), (+X, -Z), (+X, +X), (+X, +Y), (+X, +Z),
    (+Y, -X), (+Y, -Y), (+Y, -Z), (+Y, +X), (+Y, +Y), (+Y, +Z),
    (+Z, -X), (+Z, -Y), (+Z, -Z), (+Z, +X), (+Z, +Y), (+Z, +Z)]

def client_application():
    for rep in range(125):
        for corr in correlator_list:
            client_basis = corr[0]
            # Establish the entangled state and measure in the specified basis.
            outcomes[rep, corr] = create_ent(
                with=Server,
                req_type=RemoteStatePreparaion,
                measurement_basis=client_basis,
                min_fidelity=0.8)

def server_application():
    for rep in range(125):
        for corr in correlator_list:
            server_basis = corr[1]
            # Establish the entangled state.
            server_qubit = receive_ent(
                with=Client,
                req_type=RemoteStatePreparation,
                min_fidelity=0.8)
            # Perform the required rotation.
            server_qubit.rotate_basis(server_basis)
            # Measure the qubit, store the result.
            outcomes[rep, corr] = server_qubit.measure()
```

5 Supplementary note 5: Post-measurement Pauli correction

The physical layer, depending on the specific quantum platform, will deliver in general one of the four possible Bell states. With the single photon protocol we employ, the physical layer can produce either $|\Psi^+\rangle = (|01\rangle + |10\rangle)/\sqrt{2}$ or $|\Psi^-\rangle = (|01\rangle - |1\rangle)/\sqrt{2}$, see Refs. [3, 1].

We choose to offer the generation of $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ as the link layer service (the choice of the specific

state is arbitrary). In principle, one could also make the generated state a parameter of the link layer request. To deliver the desired Bell state, the link layer applies, for K-type requests (entangle and keep), a Pauli correction, by requesting a π rotation around either the X or Y axis on the client's qubit. For M-type (entangle and measure) and R-type (remote state preparation) requests, the physical layer performs a measurement in a basis prespecified—using the `PMG` command—and reports the measurement outcome, as well as which state was generated, to the link layer. The link layer can, depending on the generated state and on the chosen measurement basis, apply a classical bit flip on the client's outcome to obtain the correct measurement statistics. In particular, the link layer flips the measurement outcome of the client in the following cases: (a) State delivered $|\Psi^+\rangle$, measurement basis $\pm Y$; (b) State delivered $|\Psi^-\rangle$, measurement basis $\pm X$; (c) State delivered $|\Psi^+\rangle$ or $|\Psi^-\rangle$, measurement basis $\pm Z$.

6 Supplementary note 6: Results with and without corrections

The data presented in the main text is corrected for known measurement errors, and events in which at least one of the two devices was in the wrong charge state are removed (the CR check following the delivery of entanglement reports zero counts). While it is useful to correct for such errors in order to obtain the most faithful reconstruction of the delivered states, these errors cannot always be avoided in a real network scenario. For completeness, we report here the results first without any corrections applied, and then with only the measurement error correction applied. All the results, the raw datasets, and the software to analyze them, are available at Ref. [4].

6.1 Full Quantum State Tomography

The events in which the two devices generated 0 photon counts in the following CR check were 37 for the client and 380 for the server (out of the 4500 total). When combined, (client or server in the wrong charge state), we obtain 417 events (in zero events both client and server were in the wrong charge state). Without any corrections (tomography errors or wrong charge state), we obtain the following density matrix (which has a fidelity with the target Bell state F=0.681(16)):

$$\mathrm{Re}[\rho] = \begin{pmatrix} 0.397(9) & 0.011(9) & 0.001(7) & 0.256(14) \\ 0.011(9) & 0.058(14) & -0.005(13) & -0.007(9) \\ 0.001(7) & -0.005(13) & 0.092(12) & -0.027(13) \\ 0.256(14) & -0.007(9) & -0.027(13) & 0.452(9) \end{pmatrix},$$

$$\mathrm{Im}[\rho] = \begin{pmatrix} 0 & 0.000(18) & -0.029(9) & 0.036(9) \\ -0.000(18) & 0 & 0.010(12) & -0.002(8) \\ 0.029(9) & -0.010(12) & 0 & -0.000(8) \\ -0.036(9) & 0.002(8) & 0.000(8) & 0 \end{pmatrix}$$

Only applying tomography error correction (but not removal of wrong charge state events) yields the following density matrix (fidelity F=0.744(11)):

$$\mathrm{Re}[\rho] = \begin{pmatrix} 0.421(7) & -0.001(4) & -0.013(5) & 0.300(8) \\ -0.001(4) & 0.022(8) & -0.020(6) & -0.021(7) \\ -0.013(5) & -0.020(6) & 0.091(5) & -0.015(5) \\ 0.300(8) & -0.021(7) & -0.015(5) & 0.466(5) \end{pmatrix}$$

$$\mathrm{Im}[\rho] = \begin{pmatrix} 0 & 0.004(4) & -0.018(3) & 0.032(6) \\ -0.004(4) & 0 & 0.021(6) & 0.002(5) \\ 0.018(3) & -0.021(6) & 0 & 0.002(5) \\ -0.032(6) & -0.002(5) & -0.002(5) & 0 \end{pmatrix}$$

6.2 Fidelity vs Rate

The events in which the two devices generated 0 photon counts in the following CR check were 74 for the client and 709 for the server (out of the 10 500 total). When combined, (client or server in the wrong charge state), we obtain 781 events (there were two events in which both client and server were in the wrong charge state). Without any corrections (tomography errors or wrong charge state), we obtain the following delivered fidelities: 0.454(18), 0.540(18), 0.548(17), 0.596(17), 0.640(16), 0.674(16), 0.679(15). Only applying tomography error correction (but not removal of wrong charge state events) yields the following fidelities: 0.485(15), 0.591(14), 0.592(14), 0.652(13), 0.705(13), 0.741(12), 0.753(11).

6.3 Remote State Preparation

As mentioned in the main text, for the remote state preparation analysis, we only apply the tomography error correction for the server, while remove wrong charge state events of both the server and the client. The events in which the two devices generated 0 photon counts in the following CR check were 29 for the client and 365 for the server (out of the 4500 total). When combined, (client or server in the wrong charge state), we obtain 394 events (there were zero events in which both client and server were in the wrong charge state). Following are the numerical values that result in the plot in the main text (average fidelity F=0.853(8)):

Supplementary Table 1: Remote state preparation results with measurement error correction and with wrong charge state correction.

Client	Server			
	$\langle X \rangle$	$\langle Y \rangle$	$\langle Z \rangle$	Fidelity
Measured $\|+X\rangle$	0.634(48)	−0.123(62)	−0.004(59)	0.817(24)
Measured $\|+Y\rangle$	−0.028(58)	−0.650(45)	0.005(61)	0.825(23)
Measured $\|+Z\rangle$	−0.081(65)	−0.083(66)	0.849(31)	0.924(16)
Measured $\|-X\rangle$	−0.645(43)	0.135(59)	0.030(63)	0.823(22)
Measured $\|-Y\rangle$	0.026(65)	0.719(40)	−0.013(61)	0.860(20)
Measured $\|-Z\rangle$	0.032(58)	−0.069(58)	−0.736(39)	0.868(19)

Without any corrections (tomography errors or wrong charge state), we obtain the following prepared states, with average fidelity F=0.807(10):

Supplementary Table 2: Remote state preparation results without measurement error correction and without wrong charge state correction.

Client	Server			
	$\langle X \rangle$	$\langle Y \rangle$	$\langle Z \rangle$	Fidelity
Measured $\|x\rangle$	0.534(55)	−0.090(62)	0.009(62)	0.767(27)
Measured $\|y\rangle$	0.024(60)	−0.582(51)	−0.013(62)	0.791(26)
Measured $\|0\rangle$	−0.073(69)	−0.072(69)	0.786(42)	0.893(21)
Measured $\|-x\rangle$	−0.552(49)	0.143(61)	0.055(63)	0.776(24)
Measured $\|-y\rangle$	0.052(64)	0.623(47)	−0.018(62)	0.811(23)
Measured $\|1\rangle$	0.030(57)	−0.028(55)	−0.606(46)	0.803(23)

When only applying tomography error correction, we find an average preparation fidelity F=0.829(9):

Supplementary Table 3: Remote state preparation results with measurement error correction, without wrong charge state correction.

Client	Server			
	$\langle X \rangle$	$\langle Y \rangle$	$\langle Z \rangle$	Fidelity
Measured $\|x\rangle$	0.573(49)	−0.096(59)	0.010(58)	0.786(24)
Measured $\|y\rangle$	0.025(56)	−0.624(45)	−0.014(59)	0.812(23)
Measured $\|0\rangle$	−0.078(64)	−0.077(65)	0.843(32)	0.921(16)
Measured $\|-x\rangle$	−0.592(44)	0.153(57)	0.059(59)	0.796(22)
Measured $\|-y\rangle$	0.056(61)	0.667(41)	−0.020(59)	0.834(20)
Measured $\|1\rangle$	0.032(54)	−0.030(53)	−0.650(40)	0.825(20)

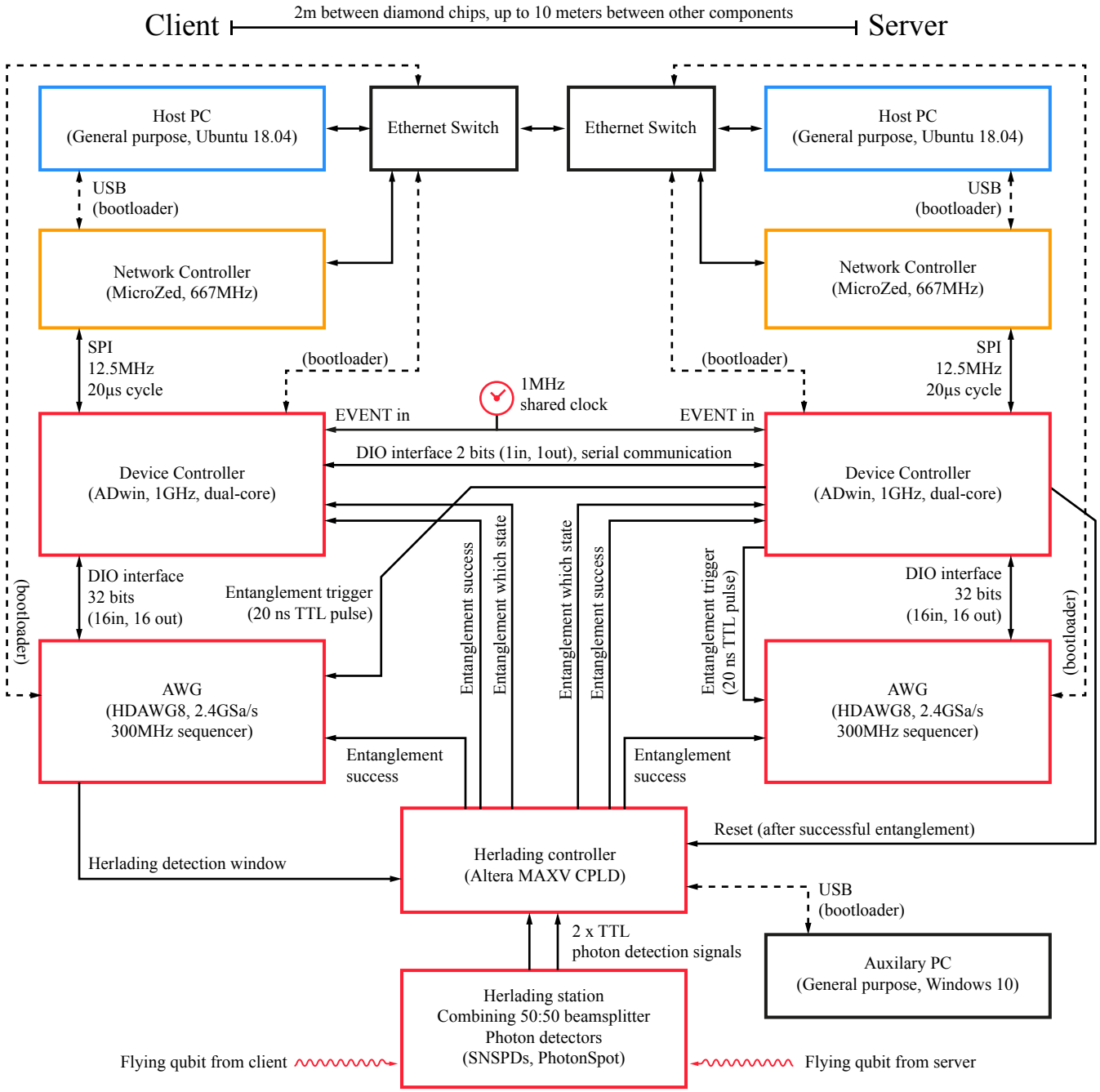
7 Supplementary note 7: NV center resonance control

The two quantum network nodes use different techniques to control the resonance of their NV centers (see Ref. [1] for implementations details). The server uses an off-resonant charge randomization strategy: when its NV center is not on resonance (it does not pass the charge and resonance check), it can apply an off-resonant (green, 515 nm) laser pulse to shuffle the charge environment and probabilistically recover the correct charge and resonance state. The server cannot get *stuck* in a non-resonance state: in a few tens of failed CR checks and green laser pulses (overall less than 1 ms) the NV center will be in resonance again.

Supplementary Table 4: List of possible outcomes of the physical layer to link layer requests.

Link layer request	Physical layer outcome	Description
INI	SUCCESS	Qubit initialization is always successful.
MSR	SUCCESS_0	Measurement outcome is $\|0\rangle$.
	SUCCESS_1	Measurement outcome is $\|1\rangle$.
SQG	SUCCESS	Single qubit gates are always successful.
PMG	SUCCESS	Updating the pre-measurement gate information is always successful.
ENT	SUCCESS_PSI_PLUS	Entanglement generation was successful, the state generated was $\|\Psi^+\rangle$.
	SUCCESS_PSI_MINUS	Entanglement generation was successful, the state generated was $\|\Psi^-\rangle$.
ENM	SUCCESS_PSI_PLUS_0	Entanglement generation was successful, the state generated was $\|\Psi^+\rangle$, and the measurement outcome was $\|0\rangle$.
	SUCCESS_PSI_PLUS_1	Entanglement generation was successful, the state generated was $\|\Psi^+\rangle$, and the measurement outcome was $\|1\rangle$.
	SUCCESS_PSI_MINUS_0	Entanglement generation was successful, the state generated was $\|\Psi^-\rangle$, and the measurement outcome was $\|0\rangle$.
	SUCCESS_PSI_MINUS_1	Entanglement generation was successful, the state generated was $\|\Psi^-\rangle$, and the measurement outcome was $\|1\rangle$.
ENT, ENM	ENT_FAILURE	Entanglement generation was attempted and failed.
	ENT_SYNC_FAILURE	Entanglement generation was not attempted because the synchronization step failed (other node is busy).
Any request	HARDWARE_FAILURE	The node has experienced a hardware problem and cannot fulfill requests.

The client, which needs to be tuned in resonance with the other node, uses a resonant strategy. When in the wrong charge state (zero counts during CR check), it applies a resonant laser pulse (yellow, 575 nm, NV^0 zero-phonon line) to go back to NV^-. To bring NV^- in resonance with the necessary lasers, it adjusts a biasing voltage applied to the diamond sample, which shifts the resonance frequencies. This process is mostly automated. However, occasional human intervention is still required when the resonance frequencies of the NV center shift too far—for example due to a charge in the vicinity of the NV center changing position in the lattice—for the automatic mechanism to find its way back. Periods of inactivity in entanglement generation are due to the jumps in the client's NV optical transitions, which then require manual optimization of the laser frequencies and/or the diamond biasing voltage—depending on the magnitude of the frequency shift, it requires tens of seconds to a few minutes to recover the optimal resonance condition.



Supplementary Figure 1: Schematic of the connections between client and server and among the various components of each quantum network node. The dashed lines represent connections used when flashing devices (boot loading), and they are not used during the real-time operation of the network stack. Not shown are additional optical and electronic components used to control the qubits, see Ref. [1] for details on the equipment.

Supplementary References

[1] M. Pompili *et al.*, *Realization of a multinode quantum network of remote solid-state qubits,* Science **372**, 259 (2021).

[2] J. Serrano *et al.*, *The white rabbit project,* https://white-rabbit.web.cern.ch/ (2013).

[3] P. C. Humphreys *et al.*, *Deterministic delivery of remote entanglement on a quantum network,* Nature **558**, 268 (2018).

[4] *Data and software supporting "Experimental demonstration of entanglement delivery using a quantum network stack",* (2021), `https://doi.org/10.4121/16912522`.