SimQ.ai

1 Why We Built It

Modern networks run on classical bits. Future networks will also carry **qubits**; single photons that enable un‑hackable encryption, teleportation of states etc.
But qubits are **fragile**. They disappear (“decohere”) as distance grows so real‑world networks will be **hybrid**:

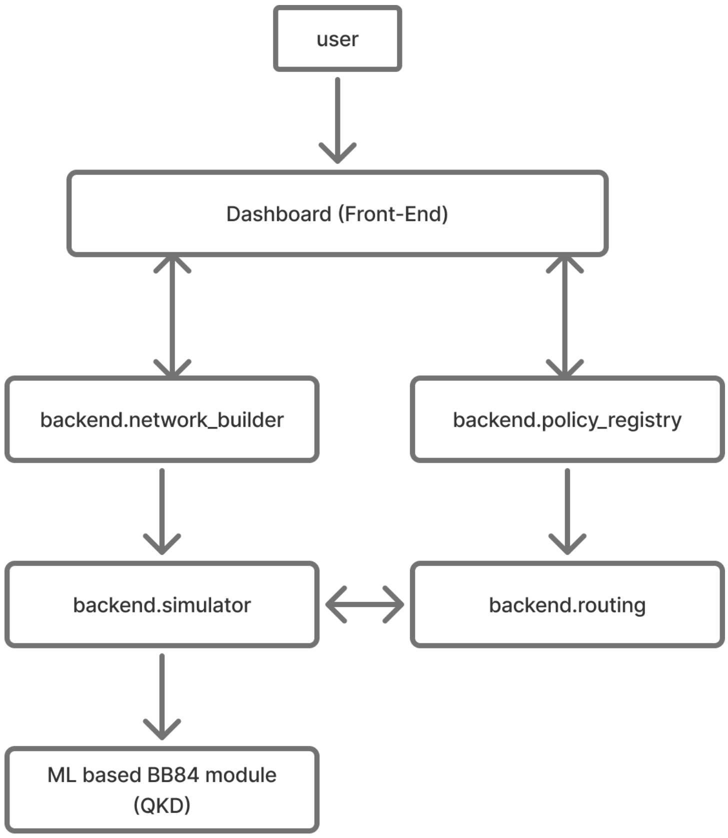
- quantum‑capable fiber
- normal fiber

Bleep Bloop lets anyone **design, simulate and debug** such a network from a web browser.
No expensive hardware, just Python + Streamlit.

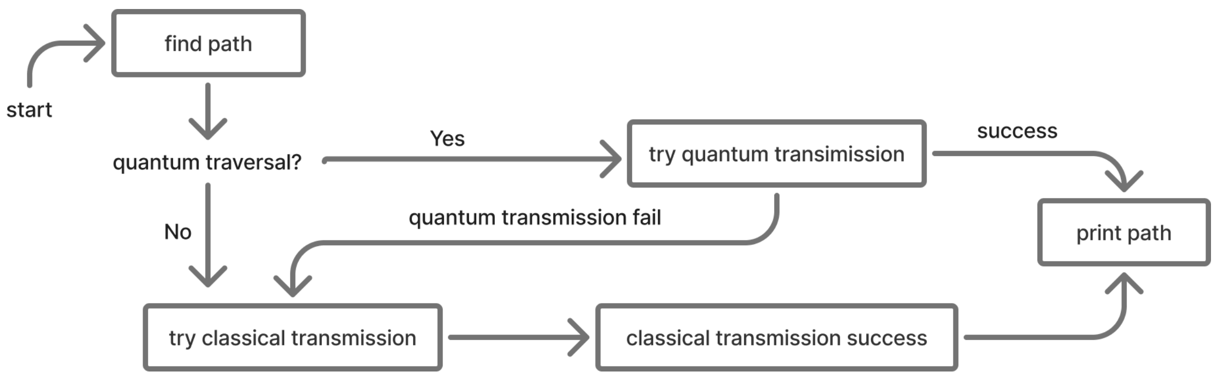
2 Feature Tour

Part	What you can do	Where it lives in the app
1. Topology Builder	Choose 10 – 500 nodes + % quantum nodes ⇒ instant graph preview.	“Network Simulator” sidebar & spring‑layout picture.
2. Link Physics	Quantum hops suffer exponential decoherence; classical hops only add latency.	simulator.py, adjustable **L (km)** slider.
3. Hybrid Routing	Tries quantum‑only path first ➜ falls back to classical **from the failure point**, not from scratch.	routing.py → hybrid() and send_message_reliable().
4. Scalability Dashboard	Auto‑draws four plots (success, hops, cost, quantum‑utilisation) as you resize the network.	Streamlit page 1.
5. Repeater Optimizer	Hill‑climb agent places ≤3 repeaters; before/after success jump shown.	repeater_env.py + demo button (Milestone 4).
6. Decoy‑State BB84 QKD	Run a full key‑exchange; optional “☠ PNS attack” aborts on decoy‑yield anomaly.	“QKD Demo” page, Part 6.
7. ML Eve Gauge	Logistic‑regression learns (QBER, ΔYield) → live “P(Eve)” %.	page 2 right column.

8. LLM Co-pilot	Describe a routing idea in plain English → Ollama writes Python → instantly selectable policy.	Sidebar of page 1.
9. Auto Insights	Bullet list explains slope & drops of every plot.	Below charts on page 1.



4 Message Flow

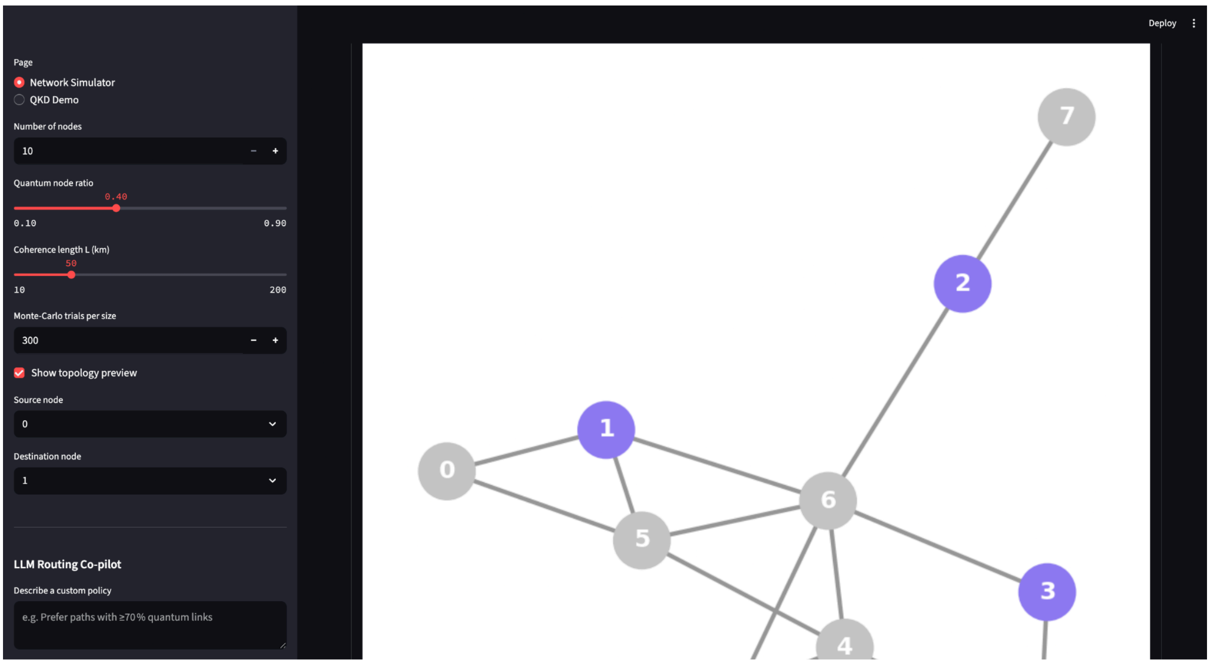


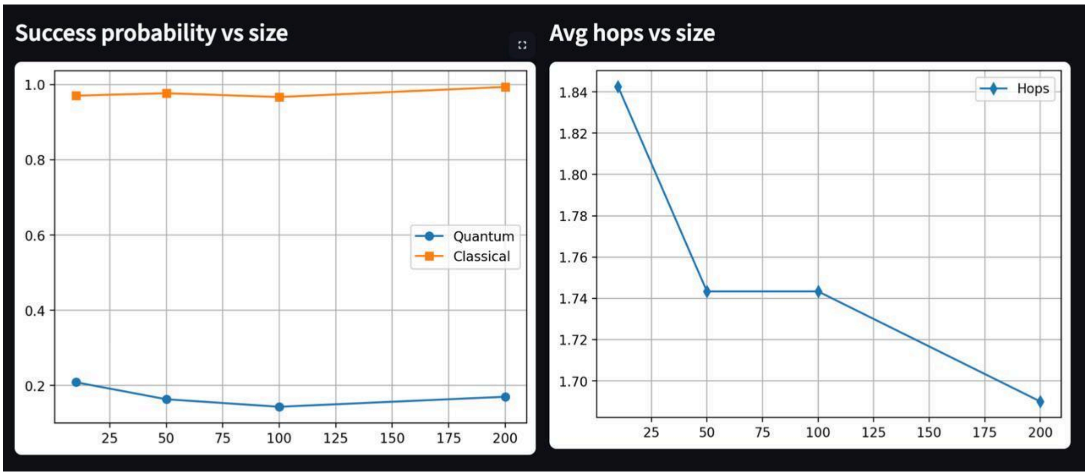
5 How we visualise decoherence

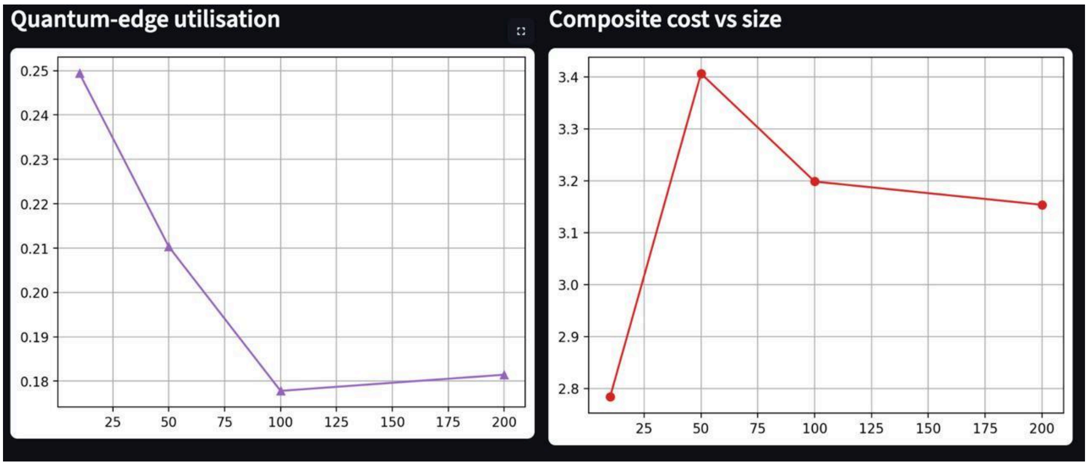
1. Slider **"Coherence L"** shortens or lengthens qubit life.
2. The blue "Quantum success" curve dives as L shrinks.
3. Composite-cost curve spikes because we add a *decoherence penalty* to every quantum hop.
4. Repeater demo resets decoherence mid-path and the blue curve jumps back up.

6 · Technology stack

1. **Python 3.10**
2. **Streamlit** for instant web GUI
3. **NetworkX** for graph maths
4. **NumPy** for Monte-Carlo physics
5. **scikit-learn** (tiny logistic regression)
6. **Ollama** (local LLM) for code generation
7. Pure **SHA-3 / HMAC / AES-GCM**—no public keys → post-quantum safe







8 Our PKI Algorithm

We enhance classic BB84 quantum key distribution (QKD), which uses single-photon polarization to create secure keys. While BB84 detects eavesdropping through quantum bit-error rate (QBER), it can be fooled by photon-number-splitting (PNS) attacks on multi-photon pulses.

To fix this, we use **decoy-state QKD**, where Alice sends pulses of varying intensity. Comparing yields of decoy vs. signal pulses helps detect PNS attacks through any yield mismatch (ΔYield).

Our improvements include:

- **Interactive decoy slider** to test photon intensities and see yield effects live.
- **Dual security check**: session ends if QBER > 11% or ΔYield > 5%, catching subtle attacks.
- **Real-time machine learning** estimates the chance of eavesdropping during each run.
- **Network simulation integration** shows how QKD behaves in real-world hybrid networks.
- **AI-optimized repeater placement** helps users design networks that balance cost, error, and security.

This hands-on sandbox lets users see threats unfold, tweak networks, and explore QKD security like never before.

9 Future work

- Swap NetworkX for **igraph** (C++ backend) → 10× bigger graphs.
- Add **quantum error‑correction chains** to repeater logic.
- Plug in a real hardware testbed (NV‑centers or Si‑photonic chips).
- UI enhancement: map overlay with real city locations.

10 Conclusion

Our project shows how the classical and quantum worlds can *co‑exist*, how fragile qubits force design trade‑offs, and how clever routing + repeaters + QKD keep data safe in a post‑quantum future.