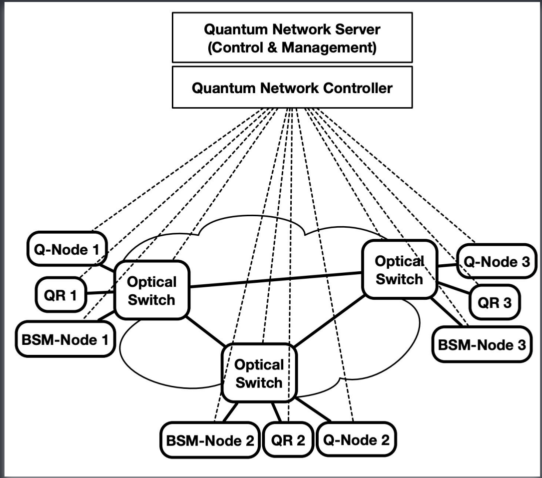
QUANT-NET Control Plane

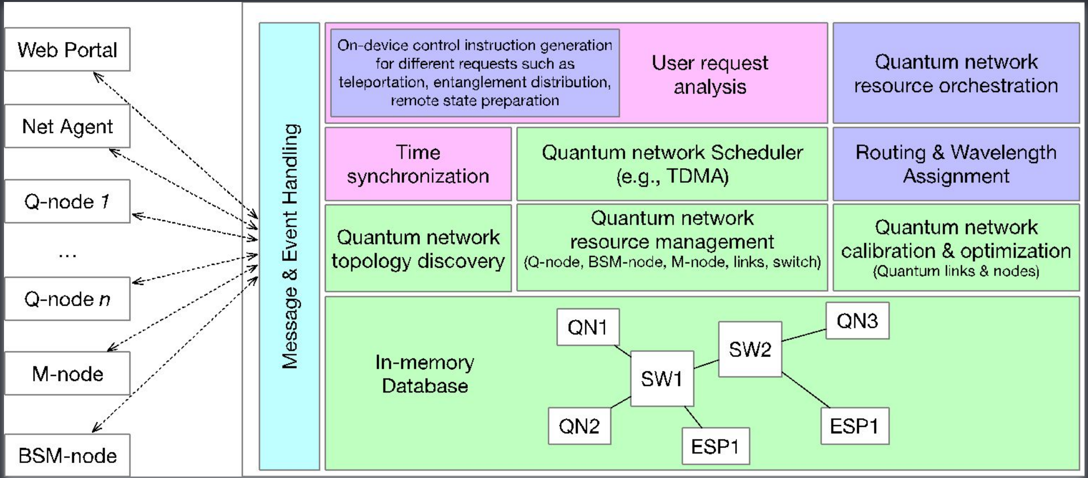
Inder Monga
Lawrence Berkeley National Laboratory
IETF 124, Montreal, Canada, November 3, 2025

Towards a framework for QIN control research

QUANT-NET Control Plane (QNCP)

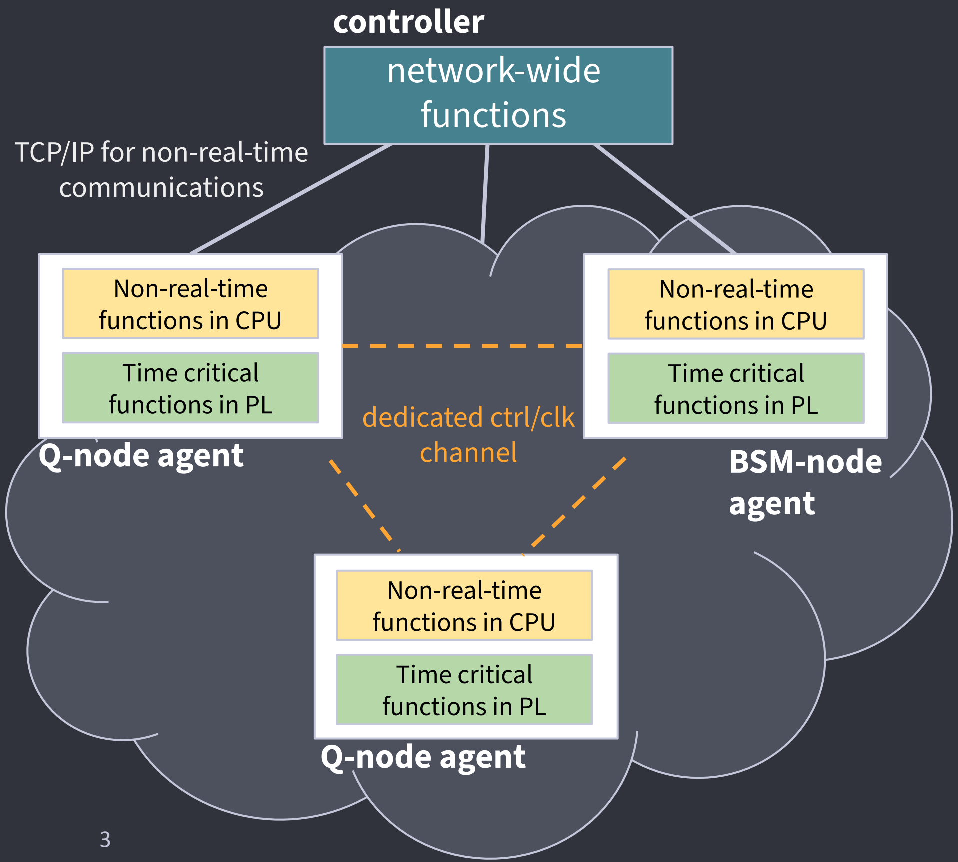
- Distributed server-agent architecture
- Interfaces from applications to physical layer
- Modular, extensible design to support rapid development





ESnet

Delineate orchestration and real-time



- Orchestration
  - Network-wide time sync
  - Calibration and automation
  - Monitoring
  - Resource state
- Real-time
  - Pulse-driven operation
  - Time-critical functions
- Protocols are required for each time domain

Provide a consistent network data model

Properties: name, ID, control interface
Node types: Q-node, BSM-node, M-node, QR-node, Switches

System

Matter qubit information: ID, type, T1, T2
Supported operations: n-qubit gates

Quantum computation resources

Matter-light interface: entanglement type, generation rate
Flying qubit: qubit type, frequency

Quantum network resources

Channels: direction, length, wavelength
Channel types: classical, quantum, clk, ...
Neighbors: channelRef, loss

Connectivity

Enable resource management and signaling

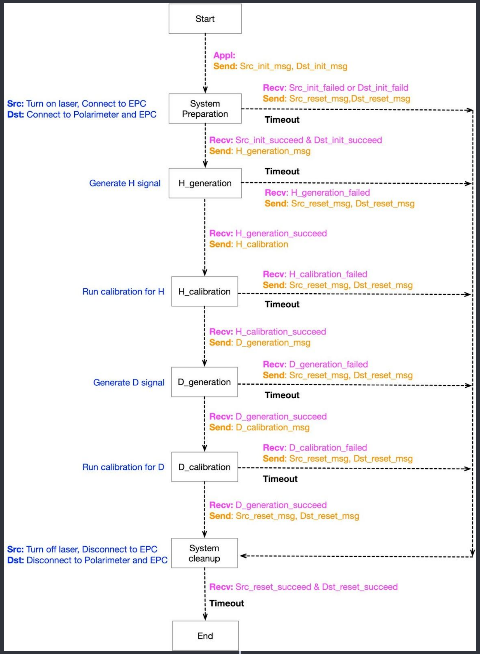
- Resource registration and discovery
  - Control agents register and send periodic updates
  - Controller maintains source of truth for the control domain

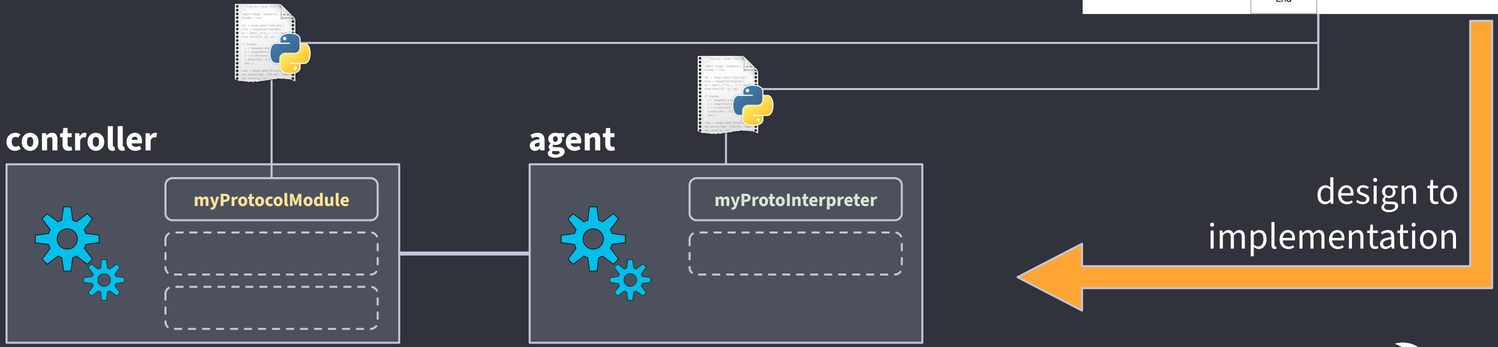
- Orchestration layer provides a communication facility between all distributed components
  - Remote procedure calls
  - Publish/subscribe
  - Write once, use everywhere schema

Allow pluggable capabilities

Develop custom protocols

- Controller plugins define protocol logic
- Agent command interpreters drive hardware
  - using devices exposed through HAL
- Controller exposes core functions
  - routing, scheduling, monitoring, etc.

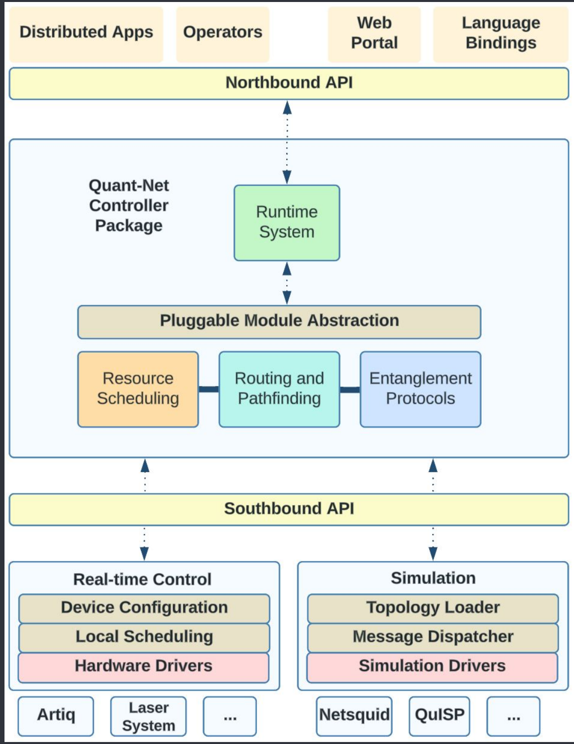




A vision for the QUANT-NET control plane

- A framework for exploring a range of quantum networking approaches
  - Include working reference implementations but within a modular, extensible software platform
  - Demonstrate real code on real hardware and create a bridge to simulation

- Create an opportunity for the quantum network community to evolve together



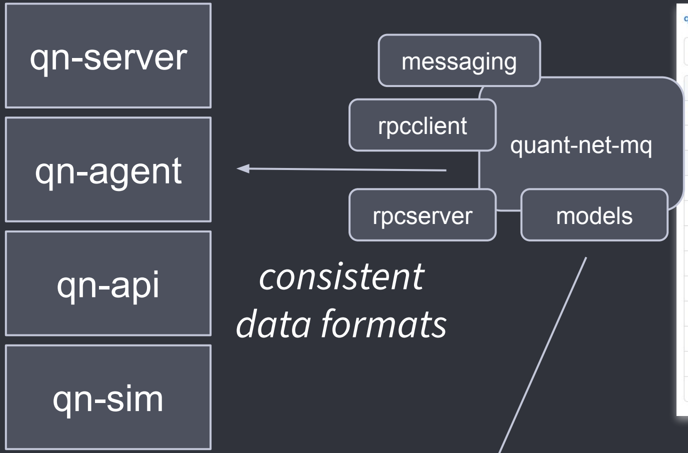
ESnet

QNCP models and interfaces

Top 3: what does the QNCP framework provide?

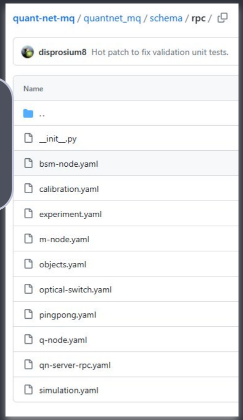
1. Software models for describing quantum network resources
2. Plugin interfaces to extend functionality
3. Hardware abstraction to develop quantum device drivers

Schema-driven framework



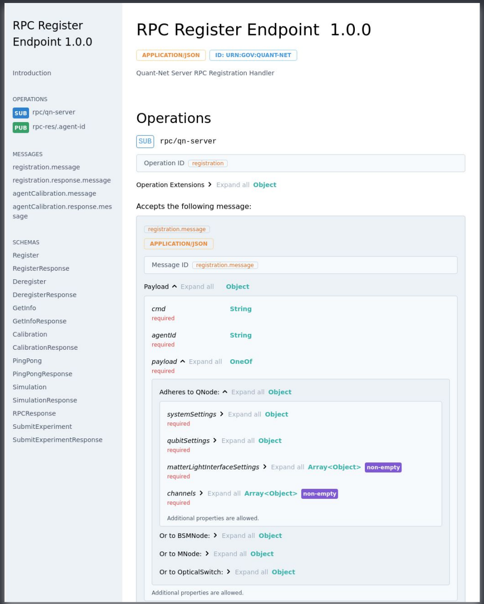
consistent data formats

<u>*source of truth*</u>




```
asyncapi generate fromTemplate \
  qn-server-rpc.yaml \
  @asyncapi/html-template \
  -o docs
```

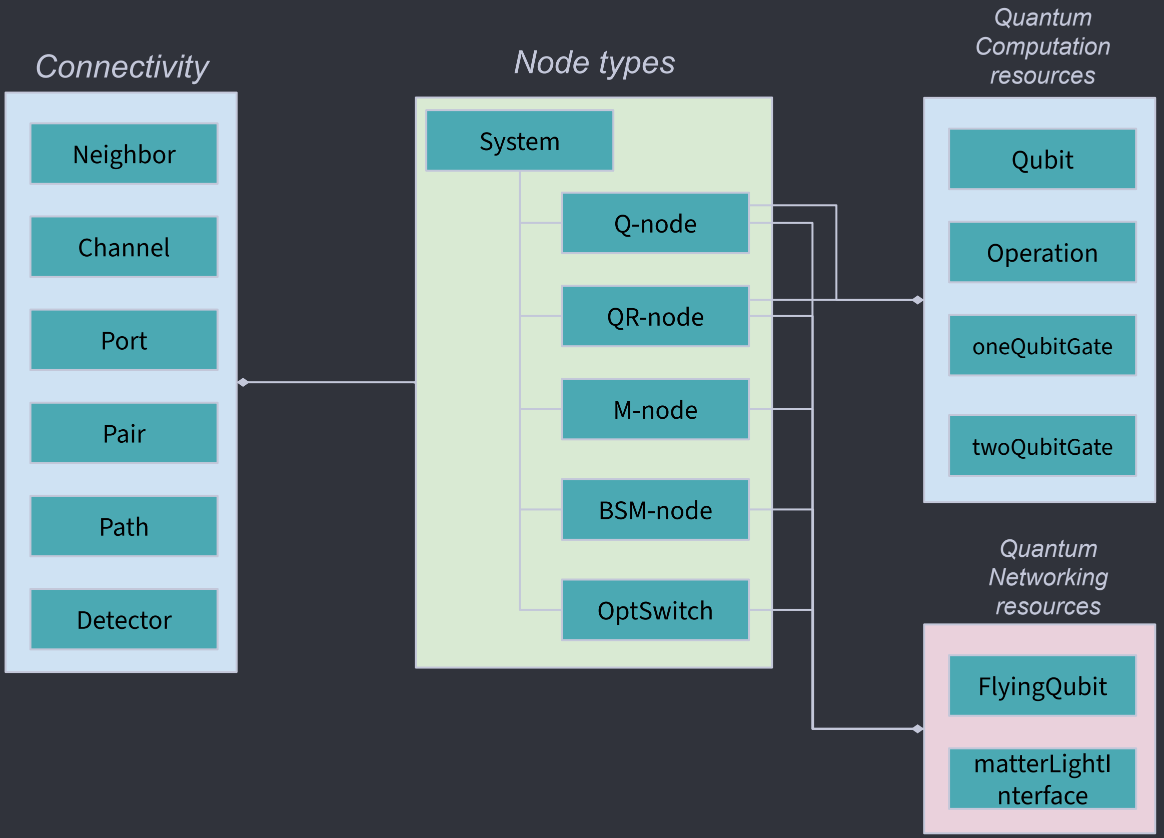
auto-generated documentation



auto-generated objects

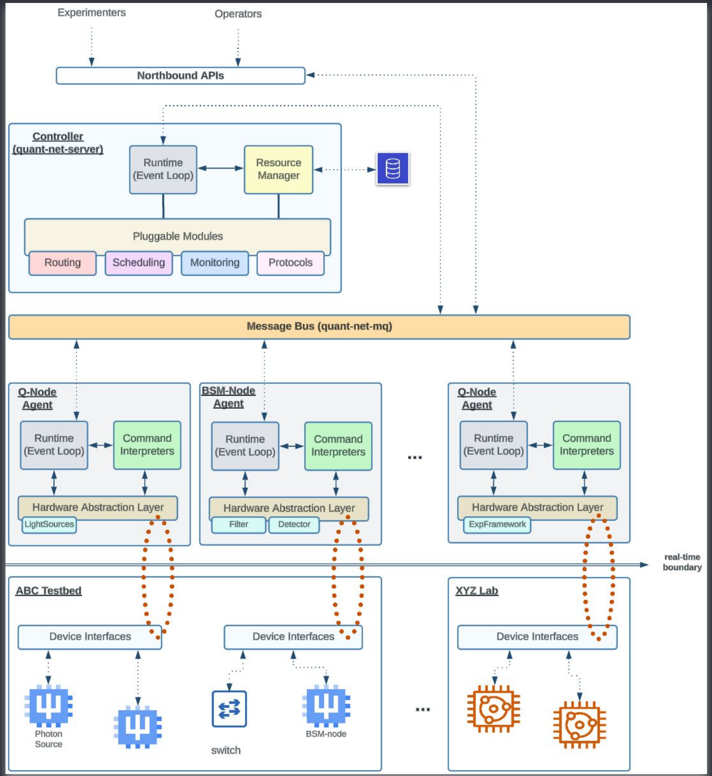
```
In [34]: from quantnet_mq.schema.models import QNode,System

In [35]: n = QNode(**qdat)

In [36]: json.loads(n.systemSettings.serialize())
Out[36]:
{'type': 'QNode',
 'name': 'q-node1.lbl.gov',
 'ID': 'LBNL-Q',
 'controlInterface': '10.0.0.3',
 'mode': 'daemon',
 'threads': 3,
 'workers': 3}
```

Quantum network resource models



Control plane component details

- Shared Message Bus
  - Communication facilities
  - Schema
- Controller Plugins
  - Routing
  - Scheduling
  - Protocols
- Agents
  - Command interpreters
  - Hardware abstraction layer



ESnet

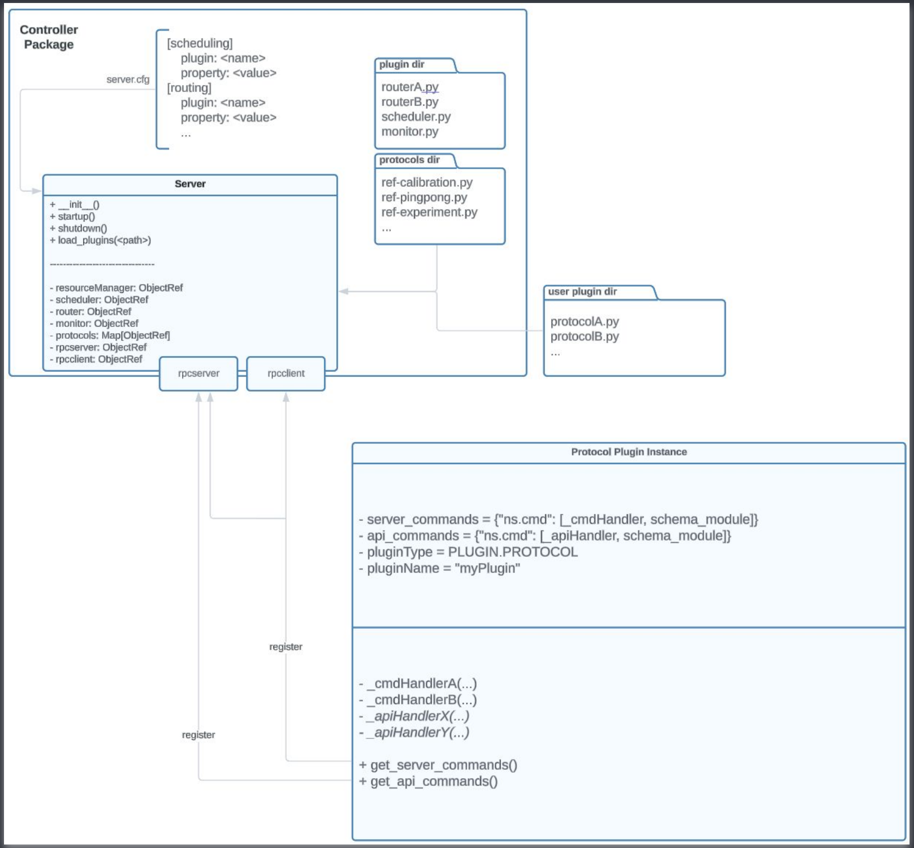
Message Bus (quant-net-mq)

- Shared module used by all QNCP components
- Provides communication layer
  - RPC-style request/response
  - Publish/Subscribe
- Provides core schema and resource modeling
- Interfaces are agnostic to underlying messaging transport
  - QNCP currently packages MQTT

ESnet

Controller plugins

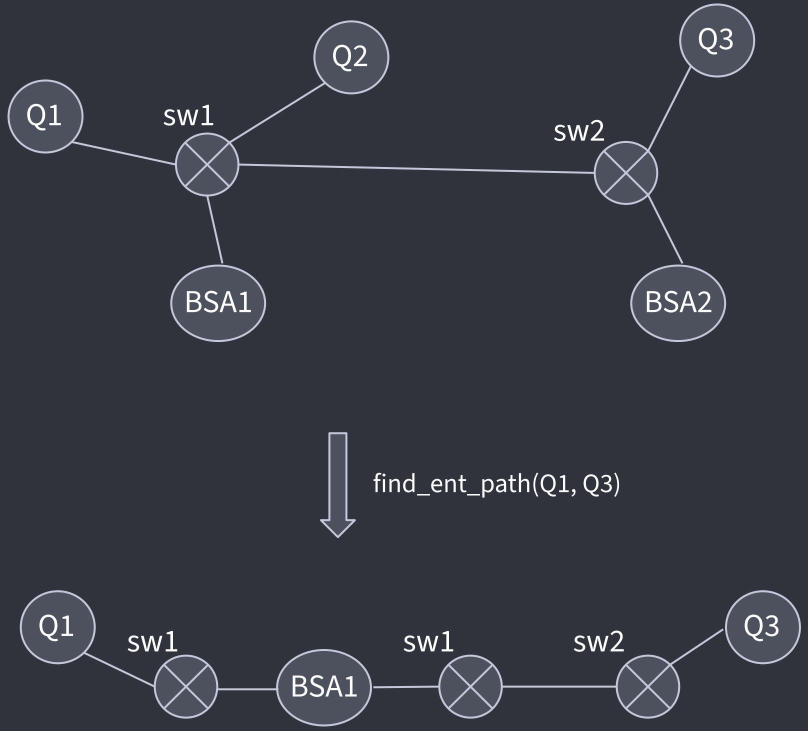
- Framework supports extensibility through pluggable modules
- Core modules (routing, scheduling) may be overridden with user-provided implementations
- Allows for arbitrary number of Protocol modules to be loaded



ESnet

Core routing plugin

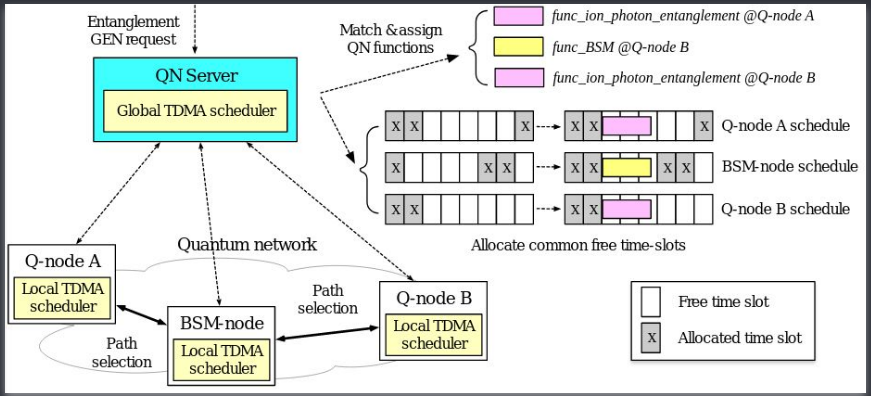
- Delivers paths over discovered control plane topology
- Reasons about entanglement generation constraints
  - e.g., including intermediate Bell state analyzers along the shortest path



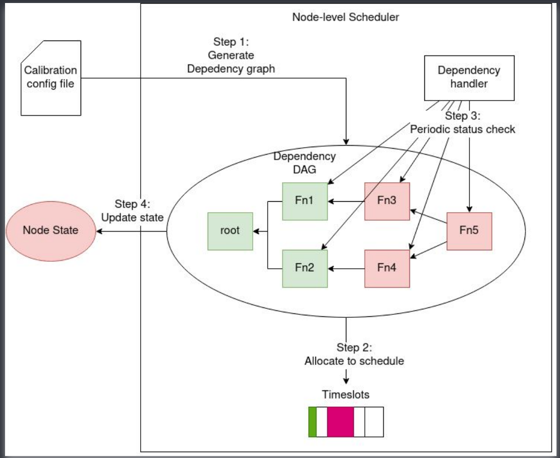
ESnet

Core scheduling plugin

- Two-level scheduler
  - Global, network-wide request handling
  - Agent-local task dependency handling
- Supports automated cycling between operational and calibration modes



Network-wide scheduling

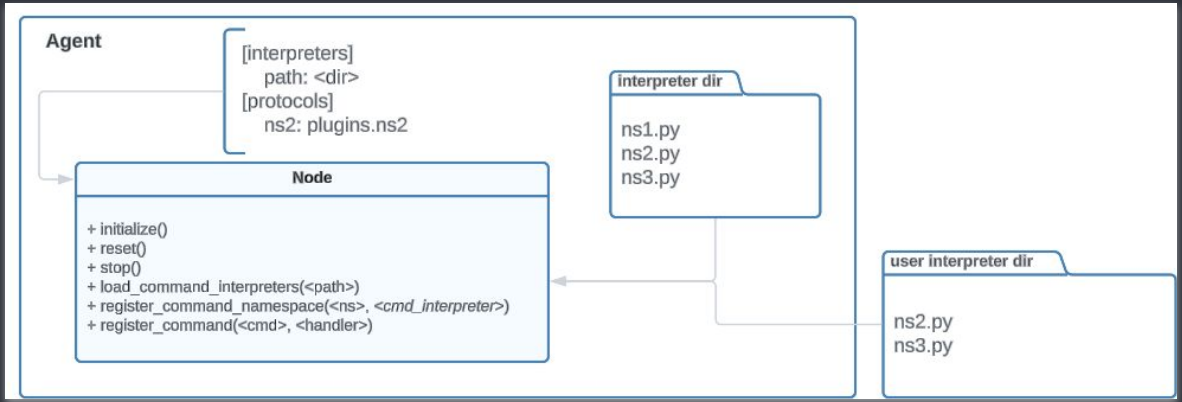


Local calibration scheduling

ESnet

Agent command interpreters

- Interpreters define the agent behavior when processing protocol messages
  - Makes use of the hardware abstraction layer (HAL) to drive the underlying hardware
- Supports user-defined plugins to be loaded at runtime
  - Can override packaged protocol namespaces for extensibility



Agent plugin modeling

Agent hardware abstractions

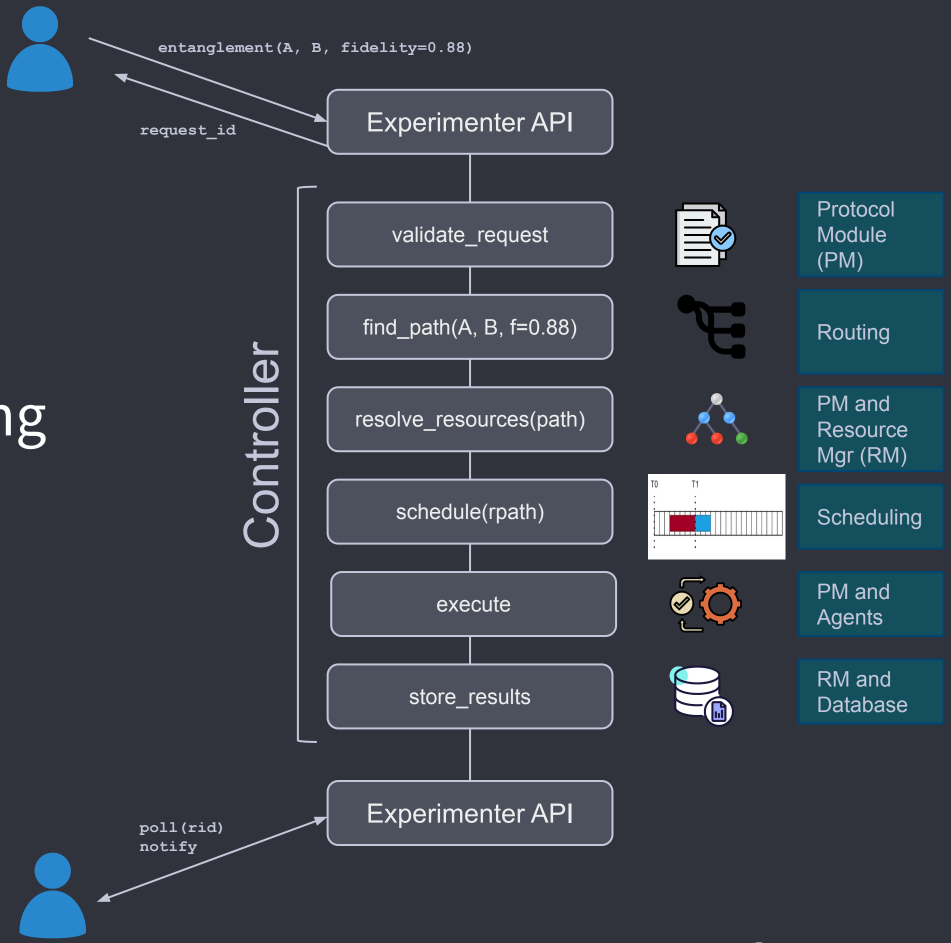
- The Hardware Abstraction Layer (HAL) provides agents with access to hardware drivers for managing real-time operations
- Device classes include:

  `LightSrc`, `Filter`, `LightMeasurement`, `SignalMeasurement`, `AnalogController`, `DigitalController`, `ExpFramework`
- Local agent configuration defines device availability

```
[devices]
[[exp_framework]]
enabled = true
type = Exp_Framework
driver = ArtiqClient
host = ::1
port = 3251

[[lightsource]]
enabled = true
type = Light_Source
driver = overlay
device = /dev/ttyACM0
baud_rate = 9600
bsm_ip = 10.0.0.7
bsm_port = 55180

[[epc]]
enabled = true
type = Filter
driver = DummyEPC

[[polarimeter]]
enabled = true
type = Light_Measurement
driver = Thorlabs
device = /dev/usbtmc0
```

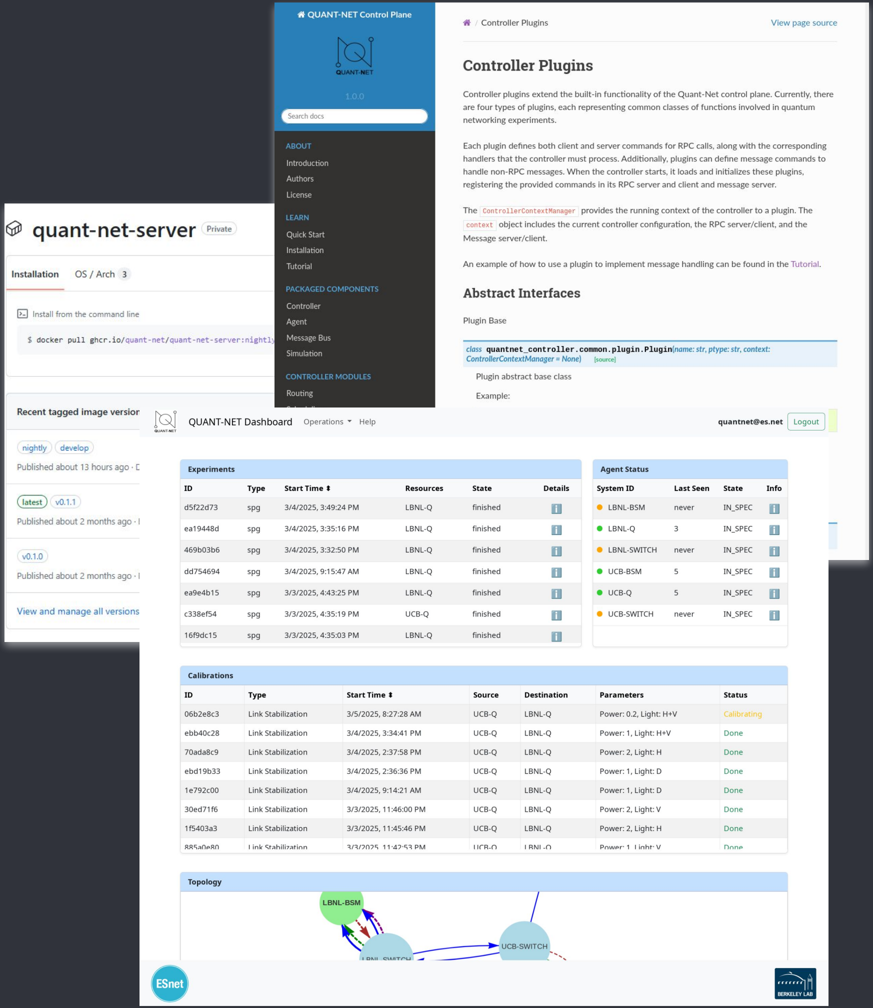
Request lifecycle

1. User submits API request
2. Request validation and pre-processing
3. Resource routing/pathfinding
4. Request resource binding
5. Scheduling
6. Execution
7. Result collection



Software deployment

- Workflows to automatically generate container images
- Software update pipelines to maintain new code on testbed control systems
- Framework documentation and tutorials
- Experimental testbed dashboards



Opportunity for IRTF

- “Rough consensus and running code”
- We have a software framework to ease the running code
- Open to quantum network testbeds to collaborate and help them implement their unique ideas as plugins that they can validate with simulation or over testbeds.
- Ease the development of code and testing of new ideas for standards