



# RMW Zenoh Workshop

## ROSCon 2025 - Singapore

**Julien Enoch**

Senior Solutions Architect  
[julien.e@zettascale.tech](mailto:julien.e@zettascale.tech)

# Zenoh





# zenoh

Pub/Sub/Query protocol that **Unifies data in motion**, data at **rest** and **computations** from embedded microcontrollers up the data centre

Provides **location-transparent** abstractions for **high performance pub/sub** and **distributed queries** across heterogeneous systems

Built-in support for zero-copy and shared memory

# Universal Abstractions

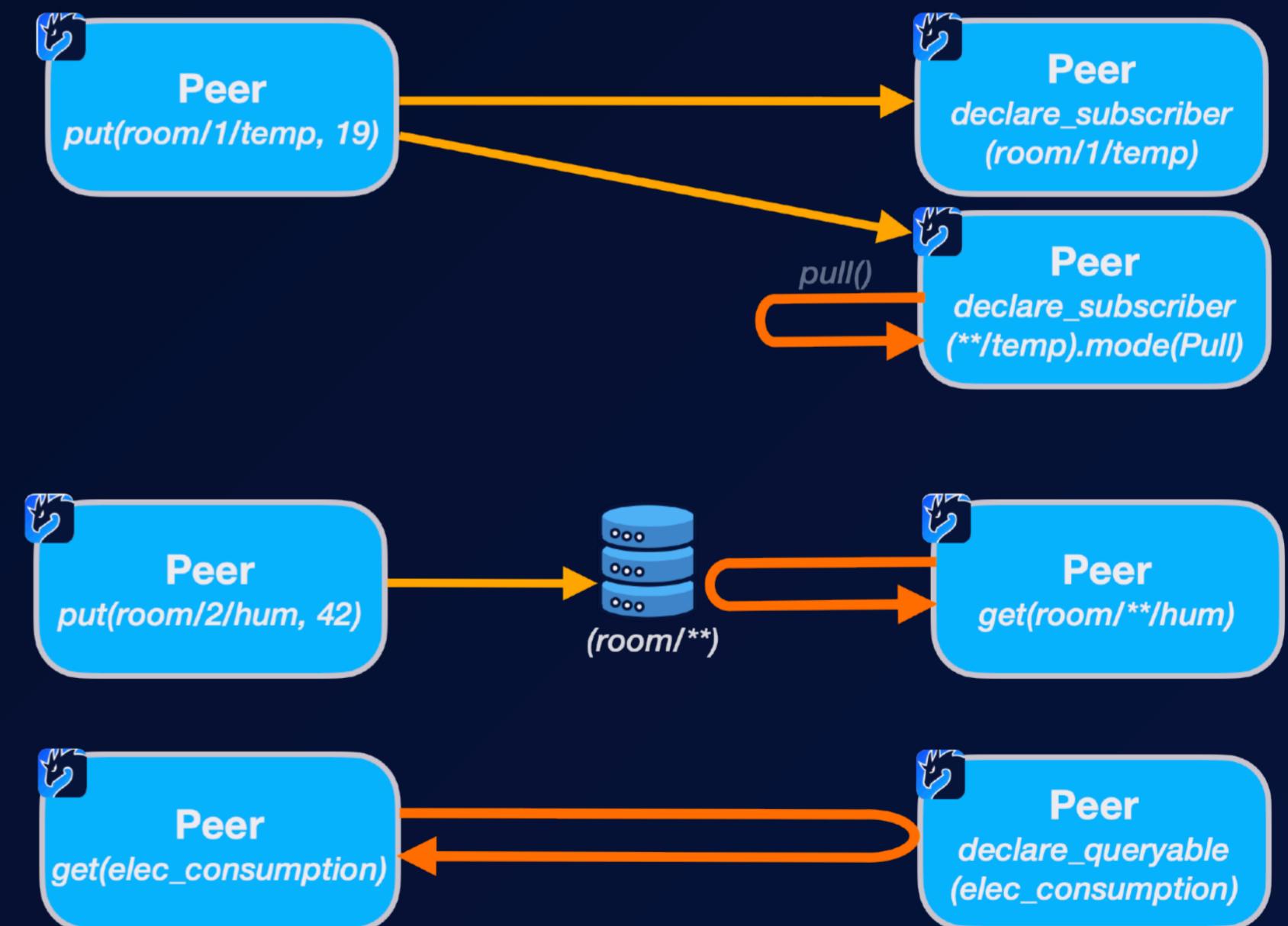
Zenoh's abstraction are **universal** since they allow to express the key patterns in distributed computing, namely:

**Publish/Subscribe.** Trivially supported by Zenoh's **Publisher** and **Subscriber**

**Remote Computation.** A **Queryable** represents a generalised computation, since it can transparently deal with replication and partitioning

**Storage.** Represented by the combination of a **Queryable** and a **Subscriber**

Additionally all these primitives enjoy location transparency by the virtue of being data-centric.



# Runs Everywhere

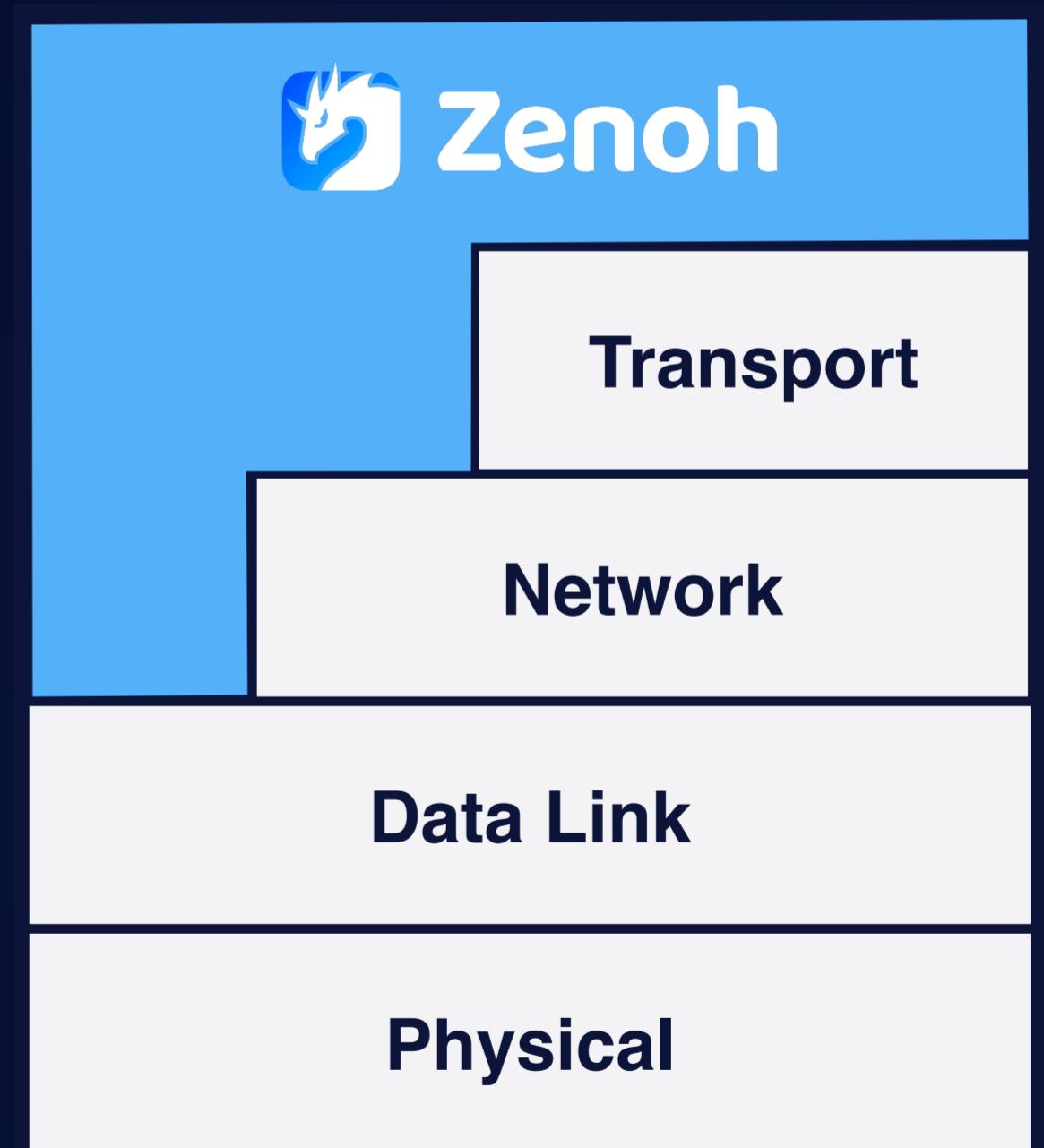
Written in Rust for security, safety and performance

**Native libraries** and **API bindings** for many **programming languages**, e.g., Rust, C/C++, Python, JS, REST, C#, Kotlin and Java

Built-in support Shared Memory and Zero Copy

Supports **network technologies** from **transport layer down-to the data link**. Currently runs on, TCP/IP, UDP/IP, QUIC, Serial, Bluetooth, OpenThreadX, Unix Sockets, Shared Memory

Available on **embedded** and **extremely constrained devices** and **networks** – 5-6 bytes minimal overhead



# Any Topology

## Peer-to-peer

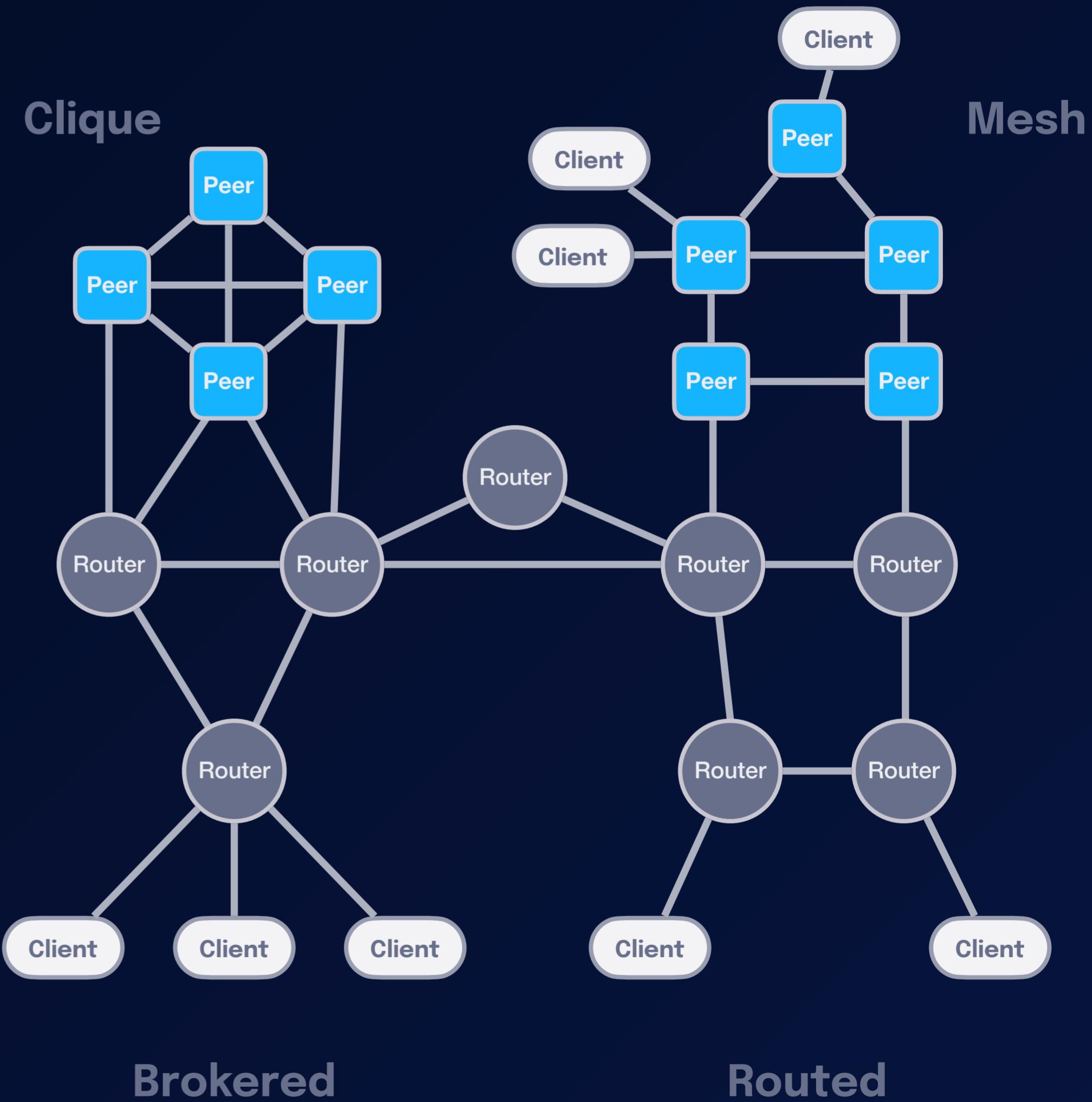
Clique and mesh topologies

## Brokered

Clients communicate  
through a router or a peer

## Routed

Routers forward data to and  
from peers and clients



# Topology in Perspective



Clique

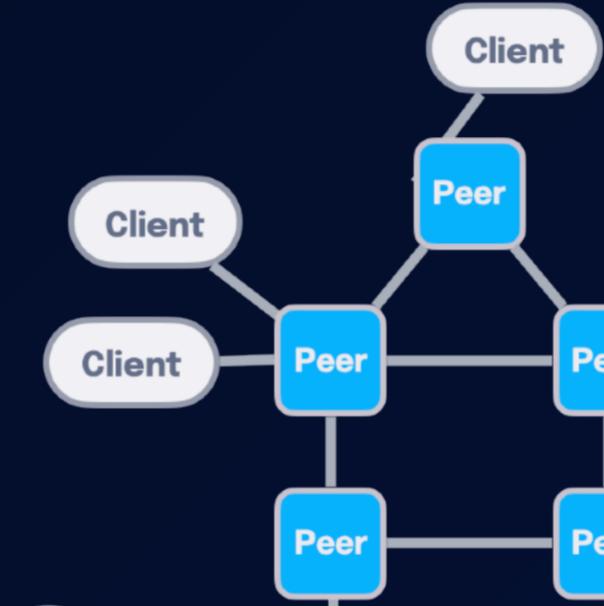


Router



Zenoh

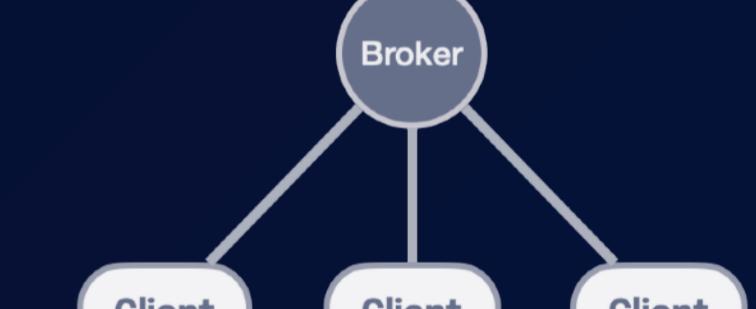
Brokered



Mesh

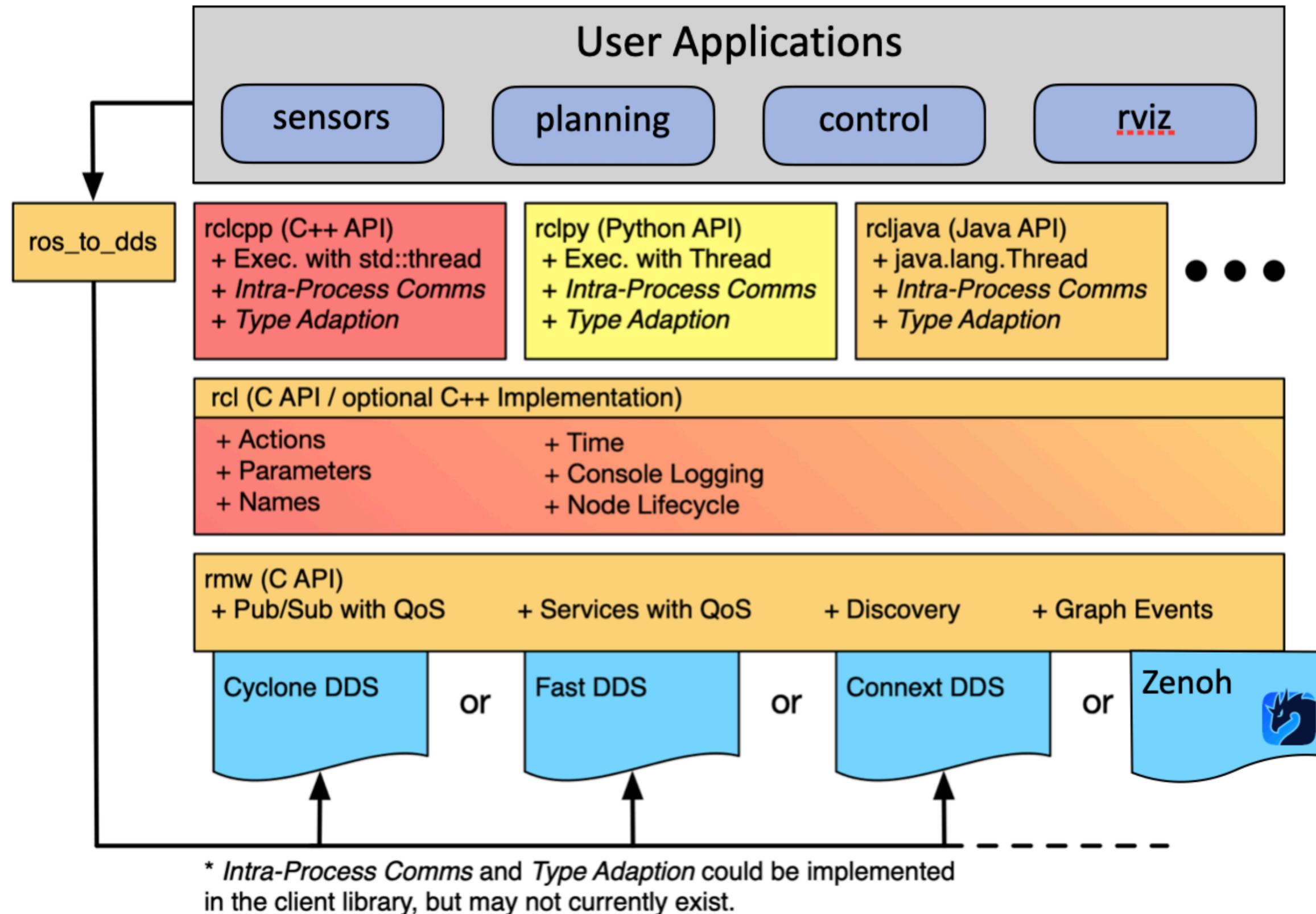
Router

Routed

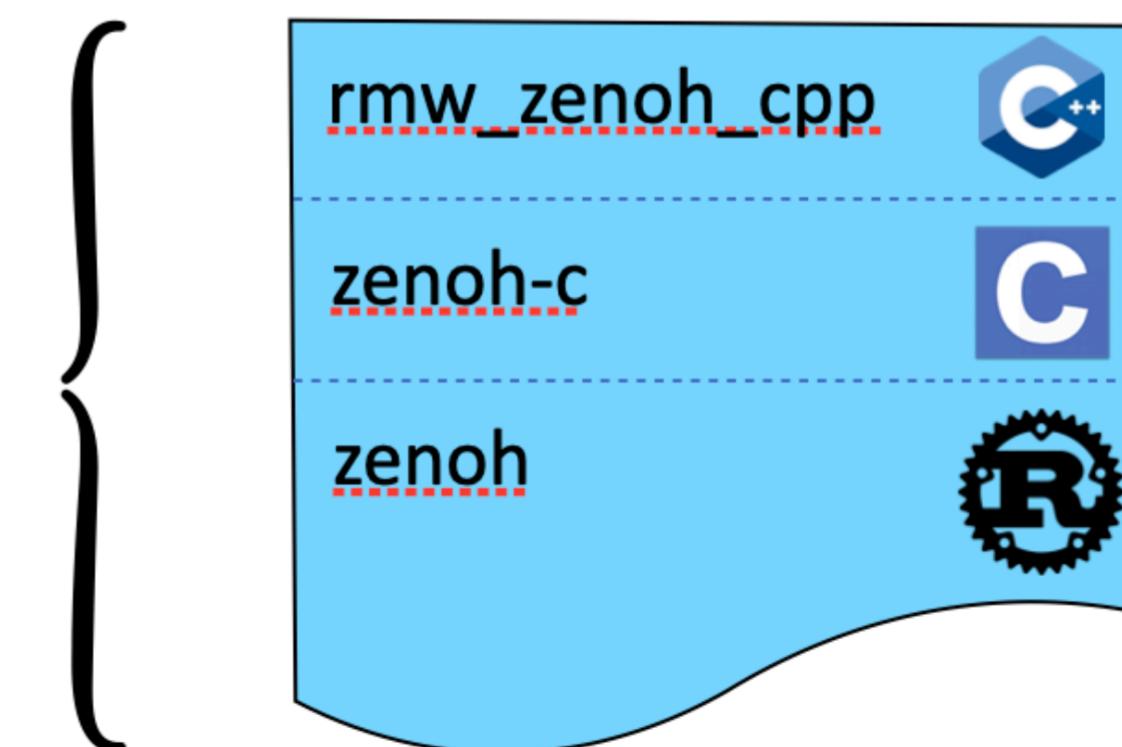


# RMW Zenoh

# ROS 2 has a modular architecture



[https://github.com/ros2/rmw\\_zenoh](https://github.com/ros2/rmw_zenoh)



# ROS to Zenoh Mapping - primitives

<b>Data encoding</b>	Still DDS <b>CDR</b> encoding via code generation: .msg/.srv/.action => .idl => serializer/deserializer code
<b>Publisher / Subscriber</b>	Zenoh Publisher / Subscriber
<b>Service Server / Client</b>	Zenoh Queryable / Querier
<b>Actions</b>	Still mapped by RCL on 3 Services and 2 Topics (thus 3 Queryables and 2 Publishers)
<b>Parameters</b>	Still mapped by RCL on Services (thus Queryables)
<b>Entities discovery and ROS Graph</b>	Zenoh Liveliness tokens (lightweight and reactive)

# ROS to Zenoh Mapping - Key Expressions

## Topics / Services:

<domain\_id>/<fully\_qualified\_name>/<type\_name>/<type\_hash>

0/robot1/chatter/std\_msgs::msg::dds\_::String\_/RIHS01\_df668c74...

0/add\_two\_ints/example\_interfaces::srv::dds\_::AddTwoInts\_/RIHS01\_e118de6b...

0/fibonacci/\_action/send\_goal/action\_tutorials\_interfaces::action::dds\_::Fibonacci\_SendGoal\_/RIHS01\_a0603060...

## TRANSIENT\_LOCAL topics:

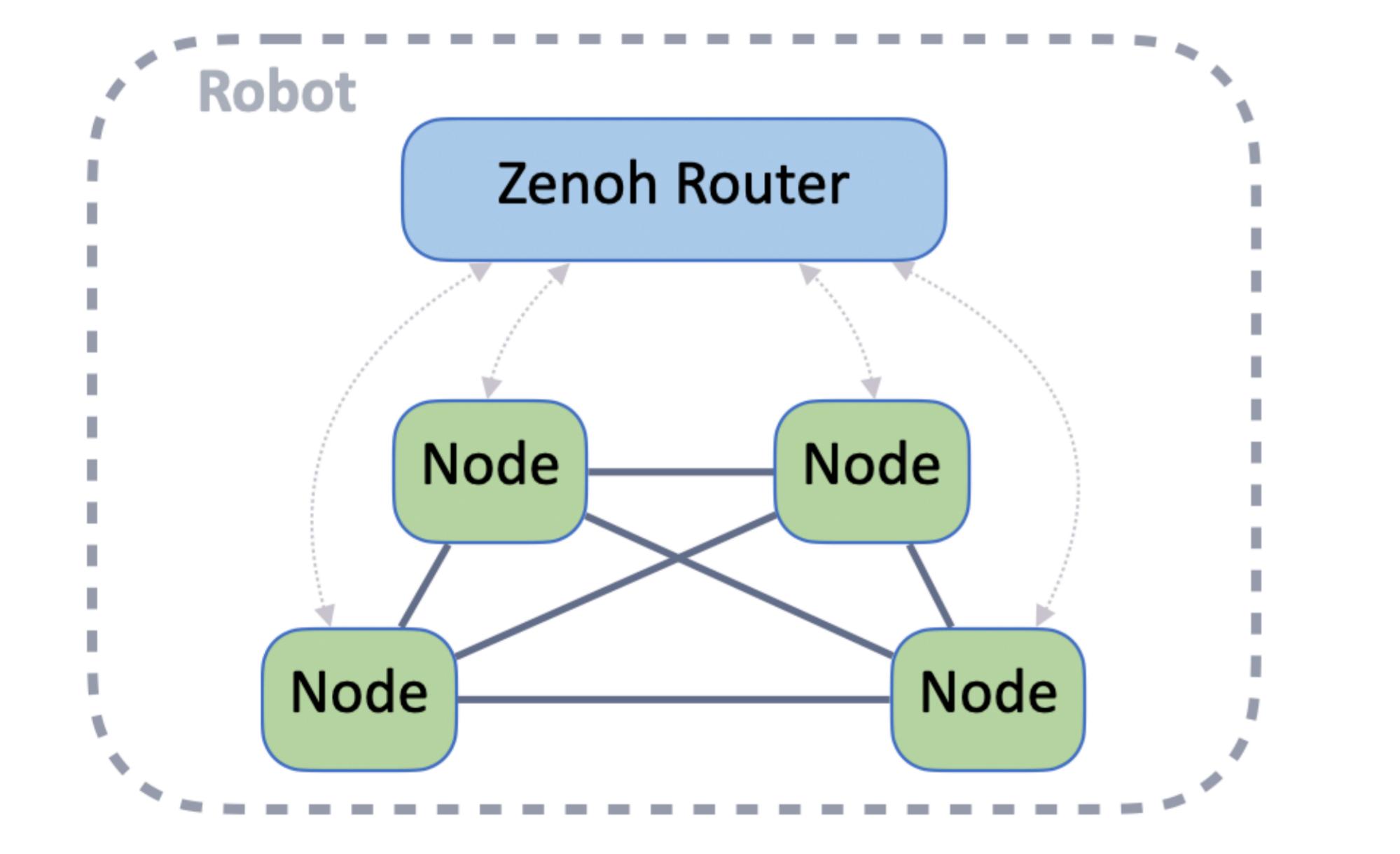
the Publisher declares an additional Queryable on

<domain\_id>/<fully\_qualified\_name>/<type\_name>/<type\_hash>/@adv/pub/<zid>/<eid>/\_

More info at [https://github.com/ros2/rmw\\_zenoh/blob/rolling/docs/design.md](https://github.com/ros2/rmw_zenoh/blob/rolling/docs/design.md)

# Zenoh Router

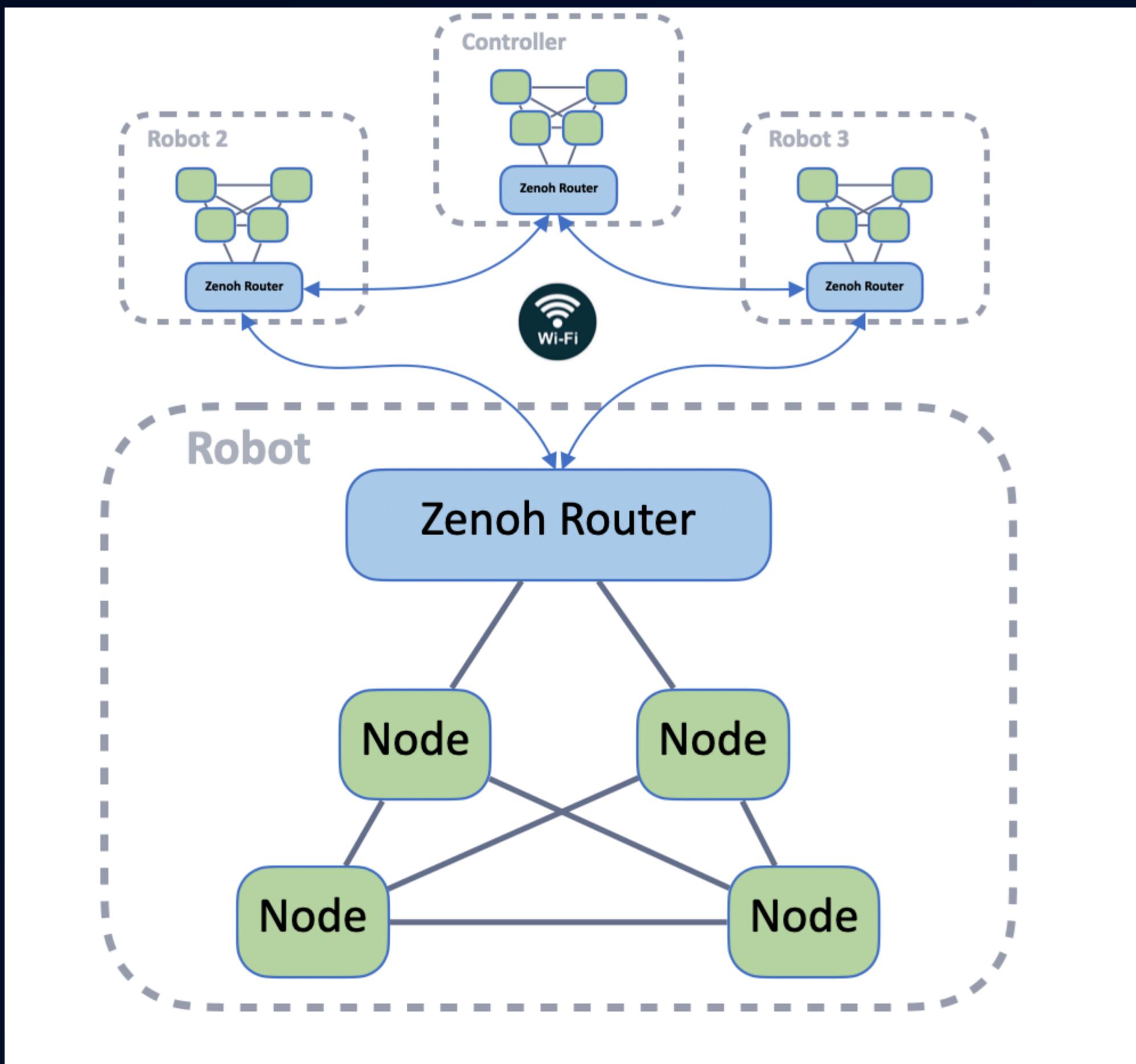
```
> ros2 run rmw_zenoh_cpp zenohd
```



## For Discovery:

- The router listen on `tcp/0.0.0.0:7447`
- Each Node connect to the router on `tcp/127.0.0.1:7447`
- The router acts as a broker for endpoints discovery
- Nodes establish peer-to-peer connections via `127.0.0.1`

# Zenoh Router



## For external communications:

- Configure the router to connect to other routers, via TCP, TLS, QUIC...
- Benefits:
  - Less connections, less overheads
  - Batching for better throughput
  - Smaller surface of attack
  - Single point to configure Access Control and Downsampling

# Installation

- Available for **Humble**, **Jazzy**, **Kilted** and **Rolling**
- Either from ROS repositories
- Either build from sources, as any ROS 2 package:

```
> cd ros_workspace
> git clone https://github.com/ros2/rmw_zenoh src/rmw_zenoh
> rosdep install --from-paths src --ignore-src --rosdistro $ROS_DISTRO -y
> colcon build --cmake-args -DCMAKE_BUILD_TYPE=Release
> export RMW_IMPLEMENTATION=rmw_zenoh_cpp
```

*Already available in your image:*

[zettascaletech/roscon2025\\_workshop](https://github.com/zettascaletech/roscon2025_workshop)

# Configuration

- 1 configuration file (json5, json or yaml) per Node and Router
- Default config files in sources under : `rmw_zenoh/rmw_zenoh_cpp/config/`
- **`$ZENOH_SESSION_CONFIG_URI`** for Nodes config
  - Most of the time nothing to change here
- **`$ZENOH_ROUTER_CONFIG_URI`** for Router config
  - External connectivities
  - TLS keys and certificates
  - Access control
  - Downsampling
  - ...



# Thank You

Patience, persistence and perspiration  
make an unbeatable combination for  
success.

