DOOCS-over-ZeroMQ

A new foundation for a 30-year-old control system

New network protocol for DOOCS: ZeroMQ

- To replace 1980s Sun RPC/ONC RPC (in the long term) ✓
- To add support for subscriptions

Opportunity: Also replace XDR serializer

- Just as old (~1984) ✓
- Supports only C data types, manual memory management

Many possibilities

- Better interface, cleaner code ✓
- User-defined structured data, see pvAccess in EPICS ≥ 7 ✓
- Lower overhead (less copying, less byte-swapping) (✓)
- Asynchronous get() and set() operations
- “Wildcard operations” that return the actual data types of the properties
- Message signing
- Compression

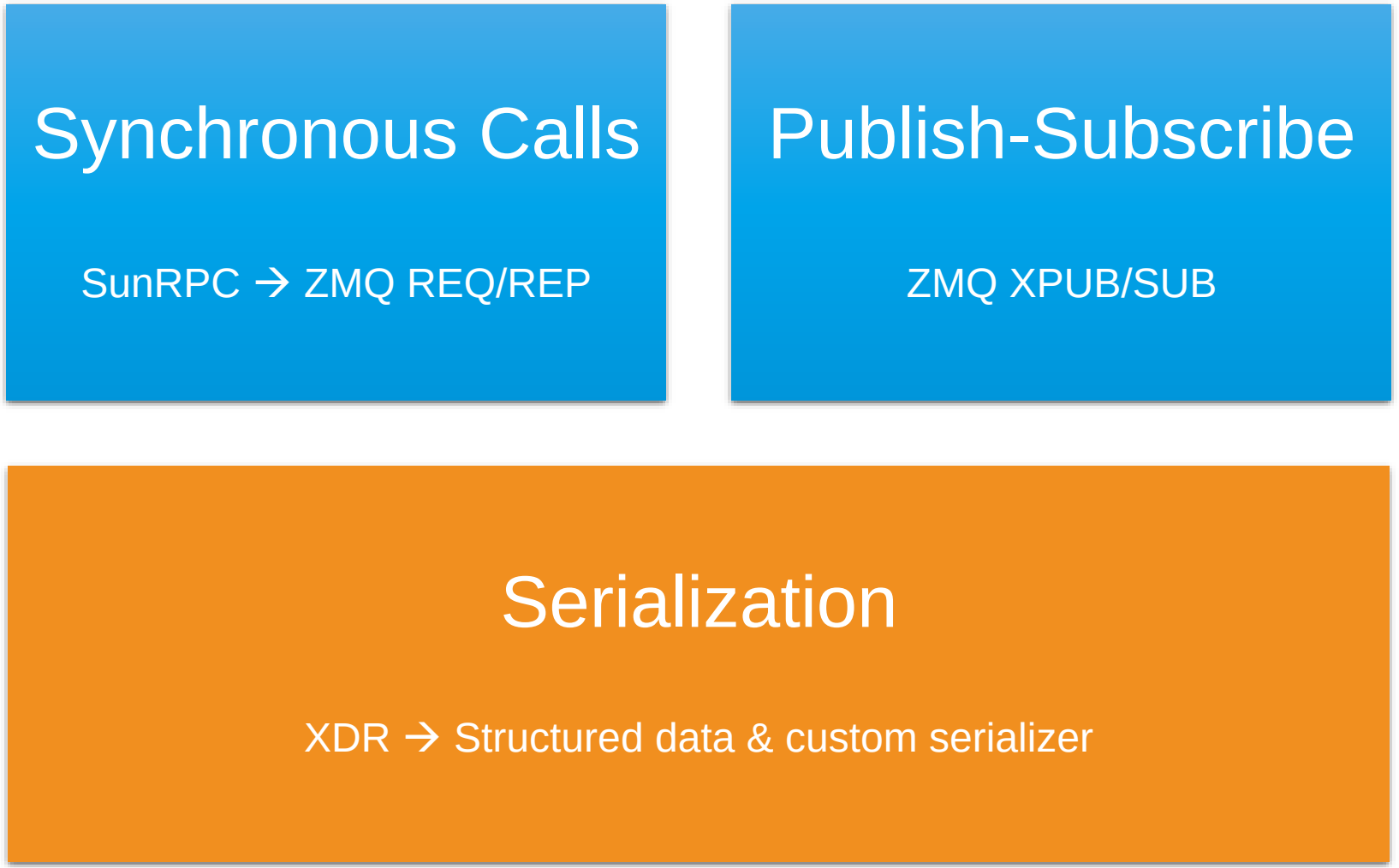
ZeroMQ in DOOCS Servers

… for synchronous and pub-sub communication

Modern DOOCS servers open 3 ports for SunRPC, ZMQ REP, ZMQ XPUB.

Communication via SunRPC still uses XDR.

Communication via ZMQ uses a custom serialization format for structured data.



doocs::Structure

A data type for structured data

A `Structure` …

- consists of an arbitrary number of `Field`s.

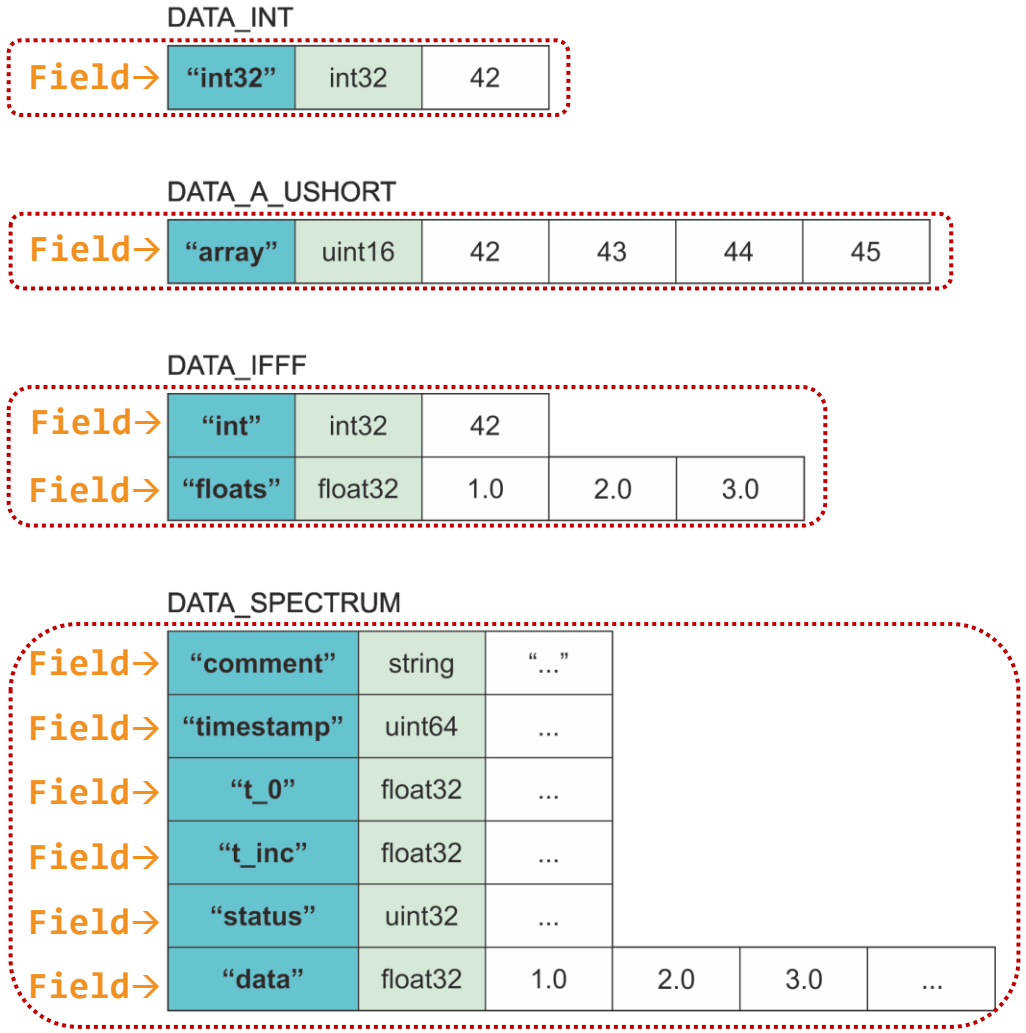
A `Field` …

- has a name,
- a type,
- a length N,
- and N data elements of that type.

Data types:

boolean, int8, uint8, int16, uint16, int32, uint32, int64, uint64, float32, float64, timestamp, event_id, string, string_view, field

Structures are self-describing and can be serialized and deserialized.



doocs::Structure

… in source code

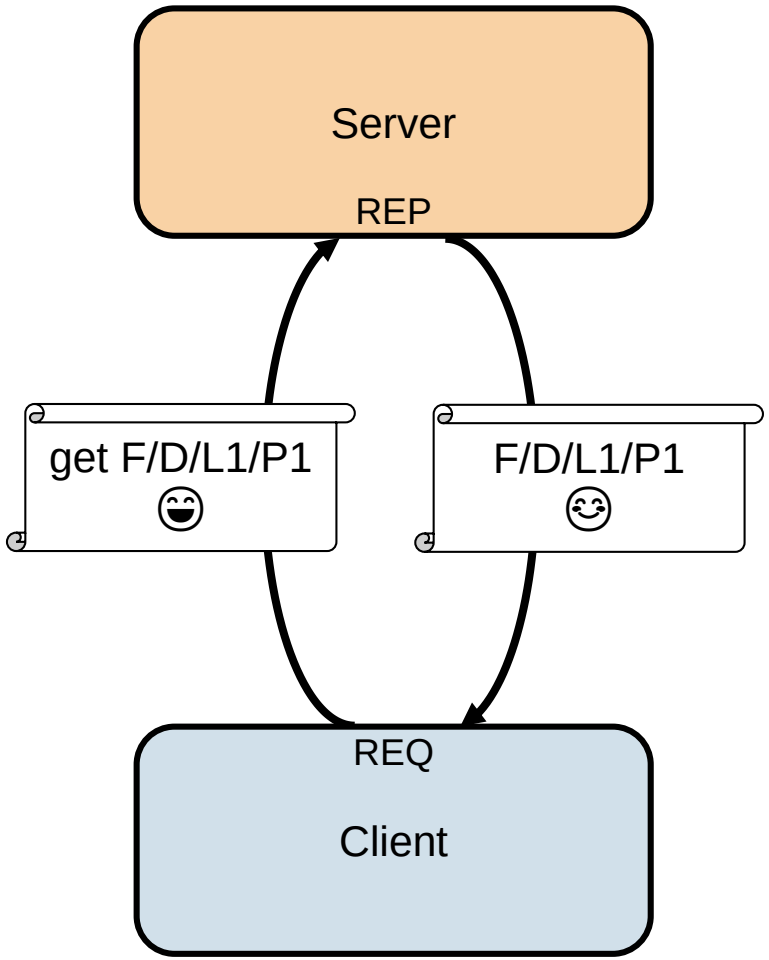
```
Structure structure{
    { "company name", "Daisy" },
    { "VAT no", 123456 },
    { "employees", Field::Vector<std::string>{ "Lars", "Manuel", "Fini" } },
    { "machines", {
        { "PETRA 3", {
            { "type", "synchrotron" },
            { "energy", 6.0 }
        } },
        { "FLASH", {
            { "type", "free-electron laser" },
            { "energy", 1.3 }
        } }
    } }
};
```

Synchronous Calls

Same Model as in Sun RPC/ONC RPC

ZMQ: REQ–REP (request–reply)

- One-to-one
- Bidirectional

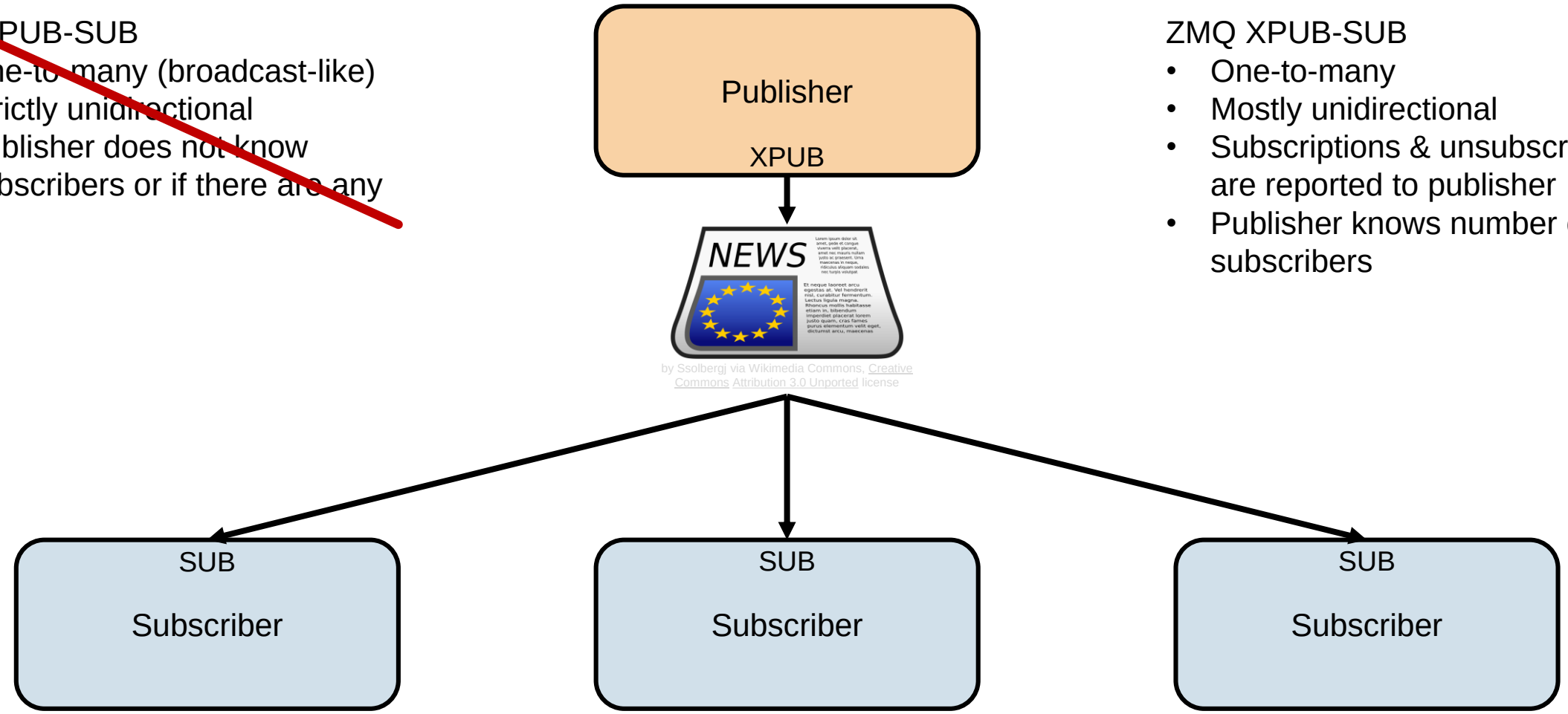


DOOCS “monitors” do this periodically in the background

Publish-Subscribe: The Idea

Avoid the Back-Channel

~~ZMQ PUB-SUB~~
- ~~One-to-many (broadcast-like)~~
- ~~Strictly unidirectional~~
- ~~Publisher does not know subscribers or if there are any~~



ZMQ XPUB-SUB
- One-to-many
- Mostly unidirectional
- Subscriptions & unsubscriptions are reported to publisher
- Publisher knows number of subscribers

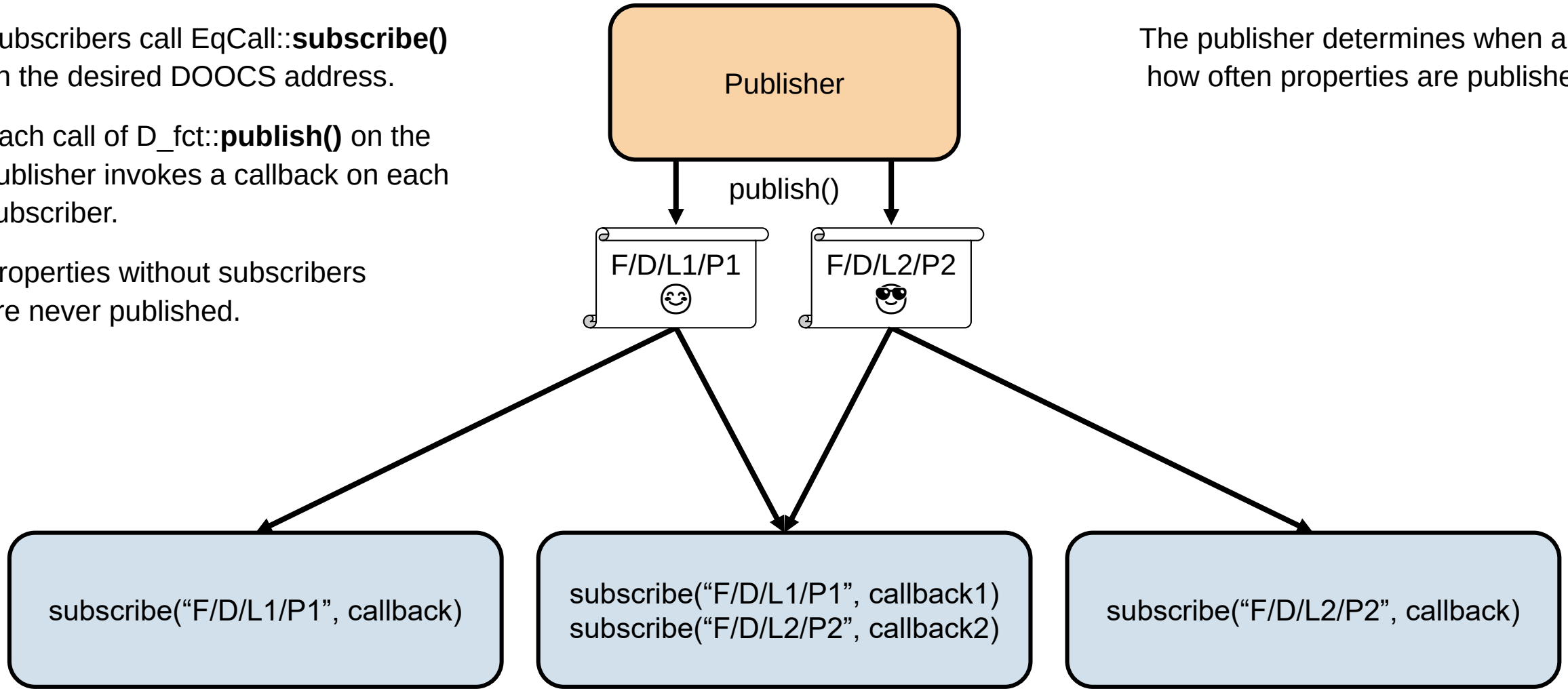
Publish-Subscribe with DOOCS-over-ZeroMQ

subscribe() and publish()

Subscribers call EqCall::**subscribe()** on the desired DOOCS address.

Each call of D_fct::**publish()** on the publisher invokes a callback on each subscriber.

Properties without subscribers are never published.



The publisher determines when and how often properties are published.

Future Idea: Asynchronous Calls

Why Wait?