SYSTEM C™

EVOLUTION DAY

OCT 16, 2025 | MUNICH | GERMANY

FSS : A proposal

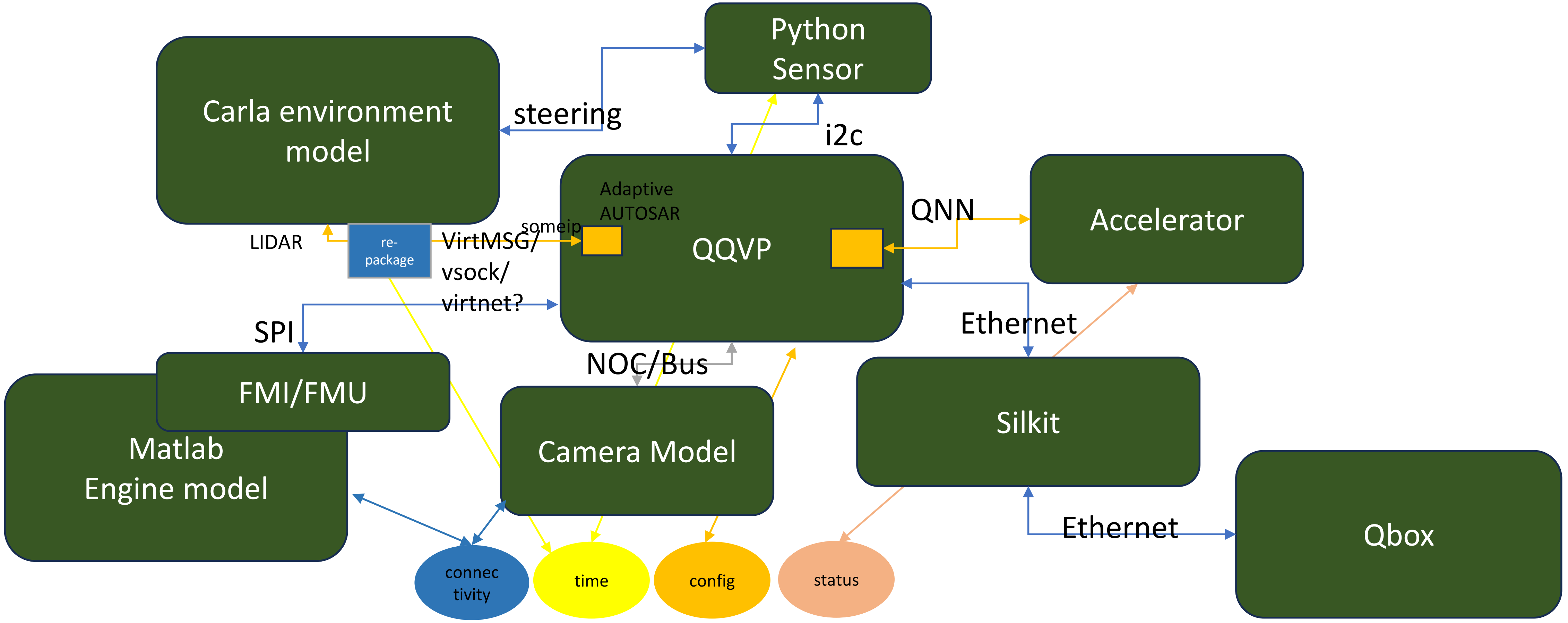
accellera

SYSTEMS INITIATIVE

Federating Simulation

- Systems of Systems.... To simulate real world (Digital twins?) we need to combine many simulators together
- No one simulator does it all
- But each simulator has different interfaces, and attempts to get them to play nicely seem to be use case specific
- We need a way to connect simulators together

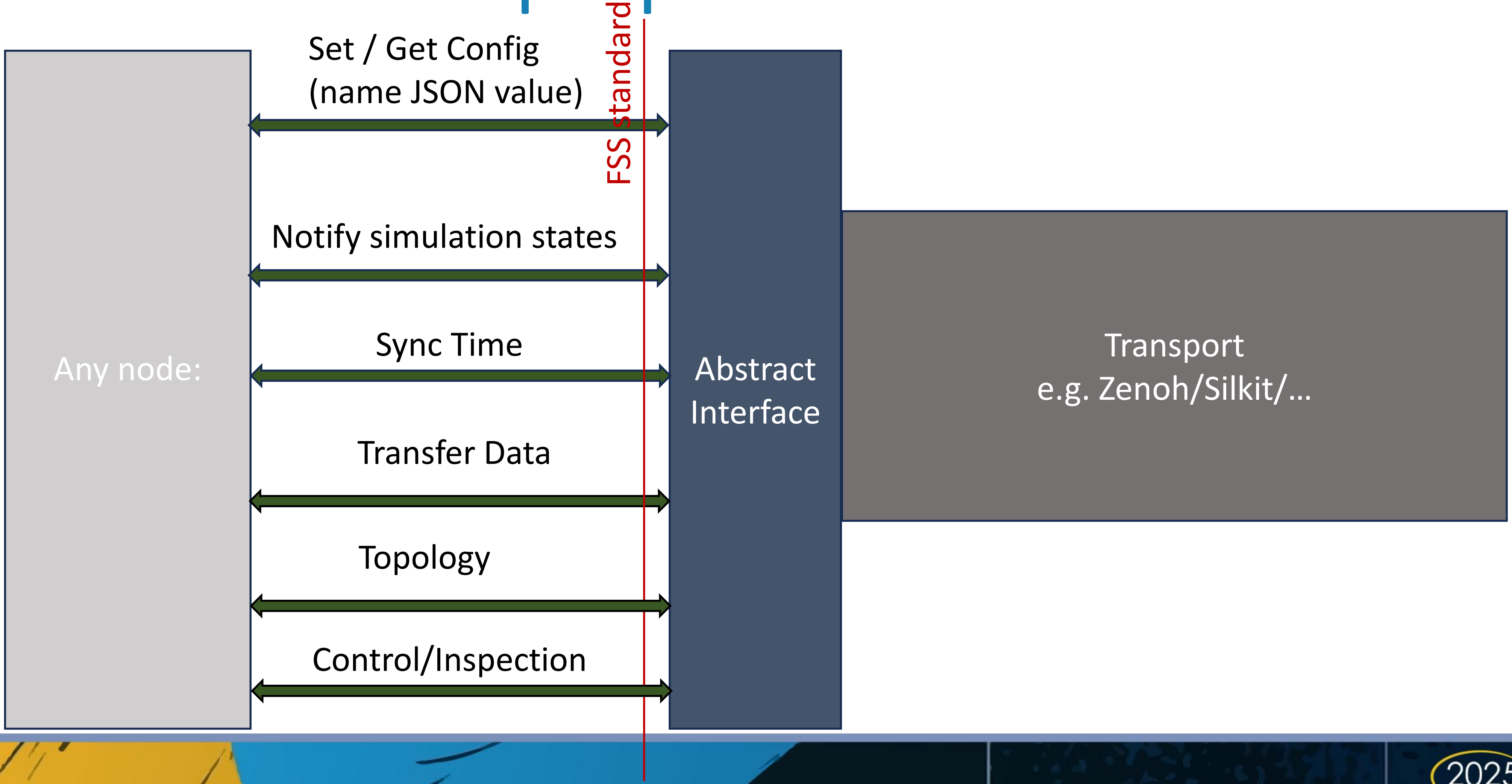
FSS goal


Python Sensor
Carla environment model
steering
i2c
Adaptive AUTOSAR
QQVP
someip
QNN
Accelerator
LIDAR
re-package
VirtMSG/ vsock/ virtnet?
SPI
Ethernet
FMI/FMU
Matlab Engine model
NOC/Bus
Camera Model
Silkit
Qbox
Ethernet
connec tivity
time
config
status

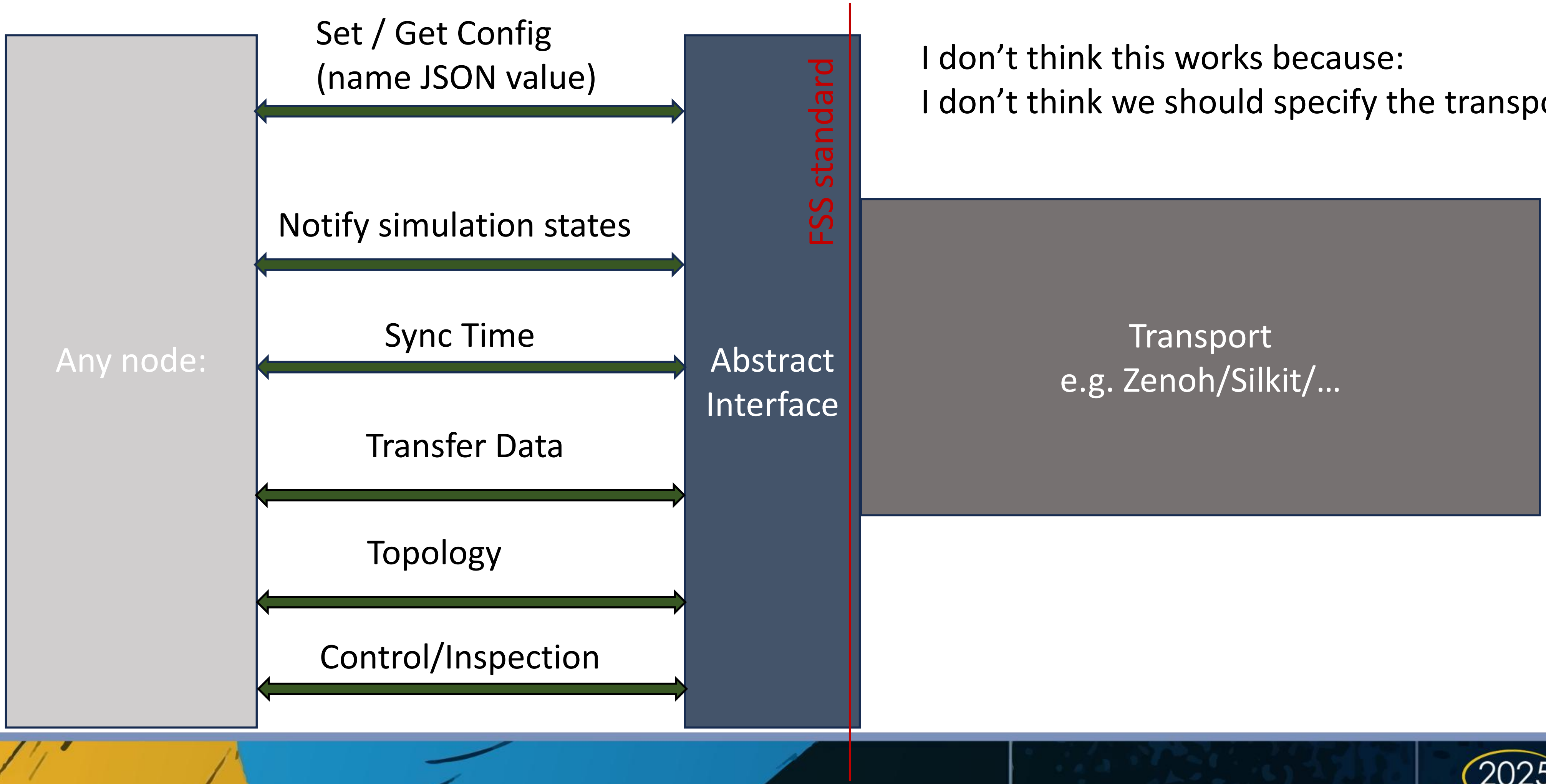
Interoperability interface needs to cover…

- Sync
- Config
- Status (Simulation state)
- Data
- Control/inspection
- Topology

FSS Interfaces : The proposal

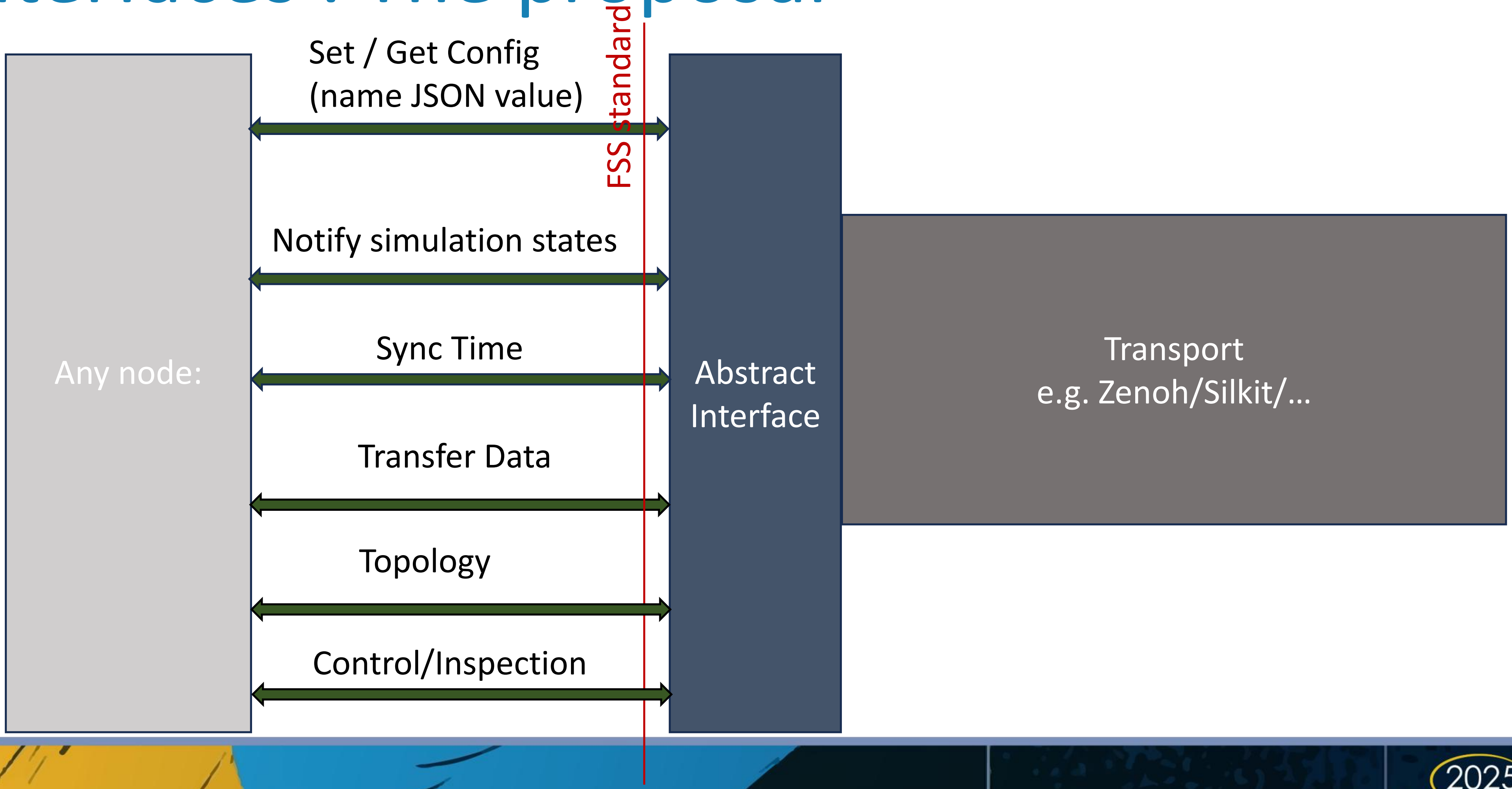


FSS Interfaces : A different proposal



I don’t think this works because:
I don’t think we should specify the transport.

FSS Interfaces : The proposal

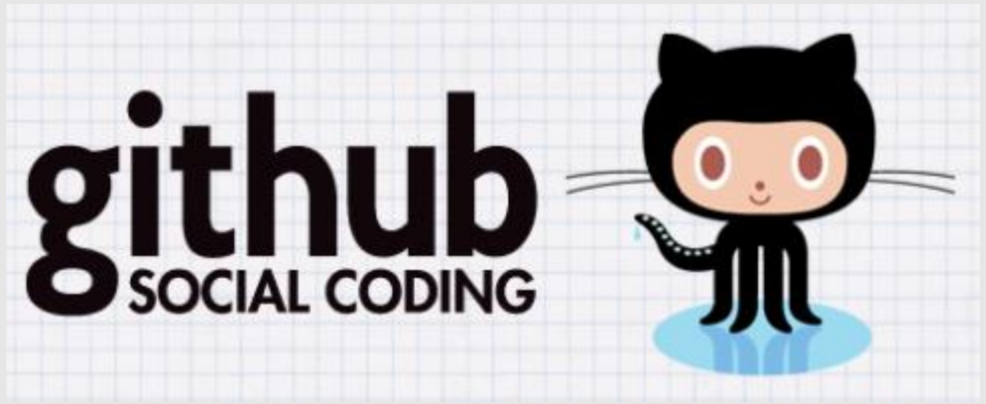


An interface in 2 parts

- SIMPLE C API

```
typedef fssBusIfHandle
(*fssGetBusIfFnPtr)(fssBi
ndIFHandle bindIfHandle,
fssString name);
```

- Semantics held on a shared public repository



This Photo by Unknown Author is licensed under CC BY-SA

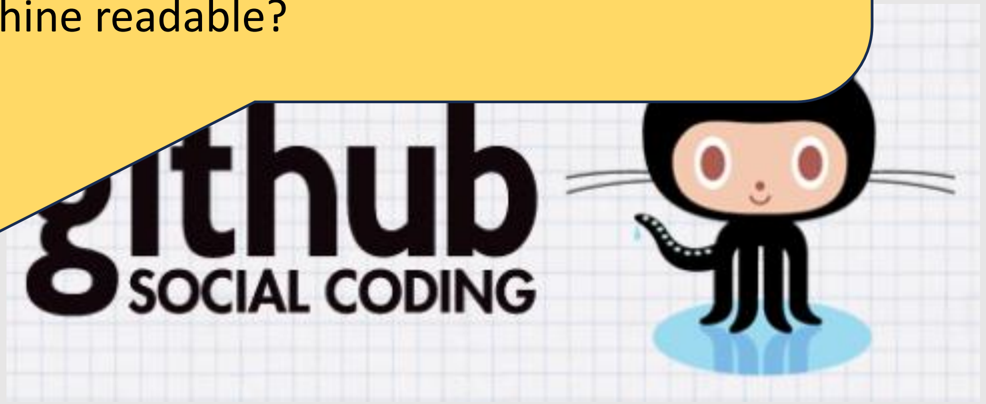
Name	Meaning
Marks.foo.bus	The special 33.5 bit foo bus mark made

An interface in 2 par

- SIMPLE C API

```
typedef fssBusIfHandle
(*fssGetBusIfFnPtr)(fssBi
ndIFHandle bindIfHandle,
fssString name);
```

WIP:
- We Don’t currently have a repo for this
- Maybe we’ll need more than “name” and “meaning”…
- Ideally I guess it would be nice if this was machine readable?



Name	Meaning
Marks.foo.bus	The special 33.5 bit foo bus mark made

General rules (Arbitrary, must be fixed)

- All names to be pre-fixed with fully qualified hierarchical name
- All names to be pre-fixed with company name/abbreviation oh author of the originator of the message
- All names to be postfixed with a message .type
- Hence:
- <company>@/foo/baa/harry/joe.sync

- All names searchable using regexp’s

Topologies

- Interfaces can be connected in any way
- Central controllers, or distributed networks

- Config databases, you may have one, or many
- Simulation events, everybody may be interested, or only 2 participants…
- Etc…

- **All is expected to be driven by configuration, and the “Configurator”**

The Top Level Component Assembler is “Special” – just not very

- “You got to start somewhere”…
- The top level component assembler owns the top level configuration database and will use that to populate the sub-modules.
- Description of overall platform will be populated into the configuration database.
- A top level assembler object will read the configuration database and build the requested components
- Assembler will pre-populate much of the configuration database
- Assembler will provide “names” to nodes

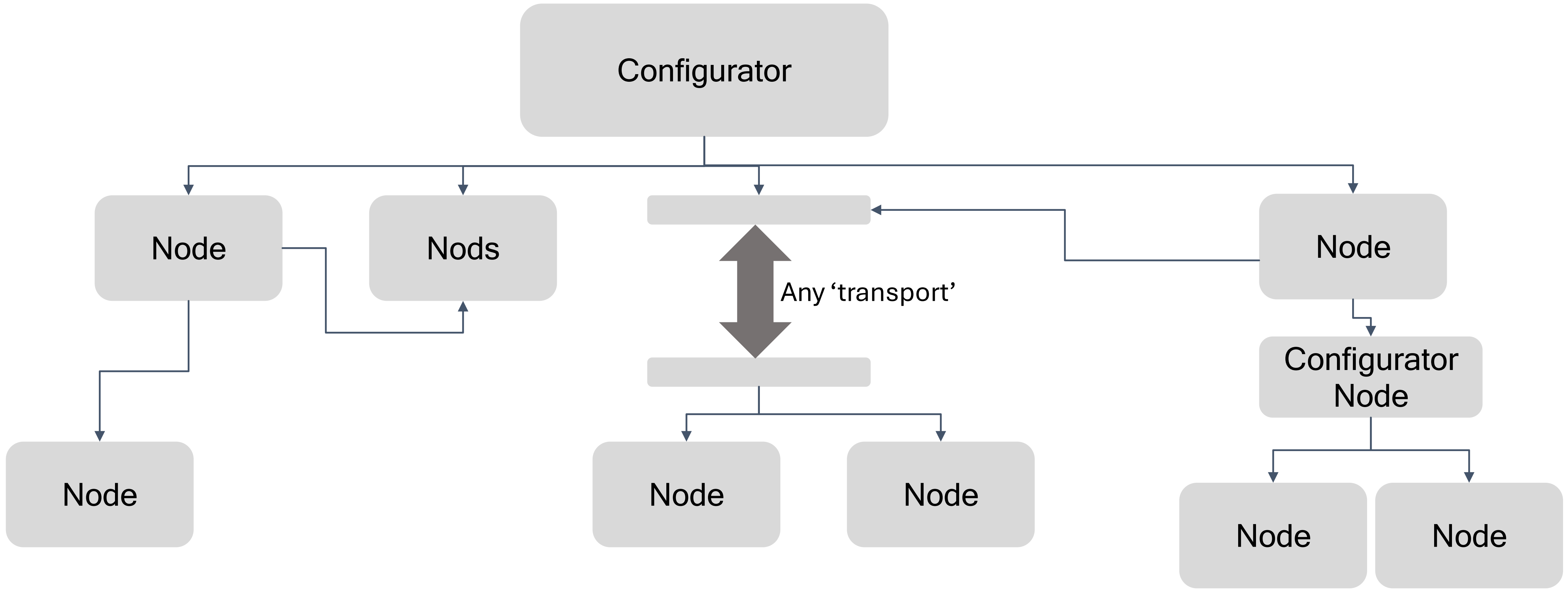
Alternative approach…

Simulators could just ‘exist’ and find each other, and connect.

Nice idea – but the reality is a PITA !

Start and stop each simulator independently? Yuk… Surely half the reason for this standard is to have a “standard” way of coordinating that!

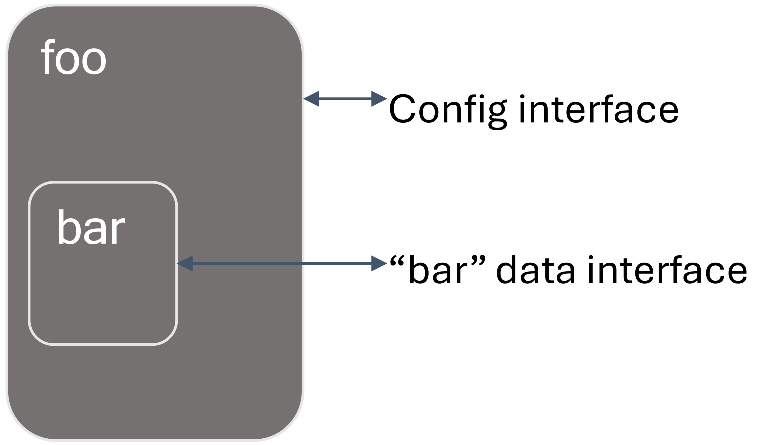
Hierarchy of a federated simulation



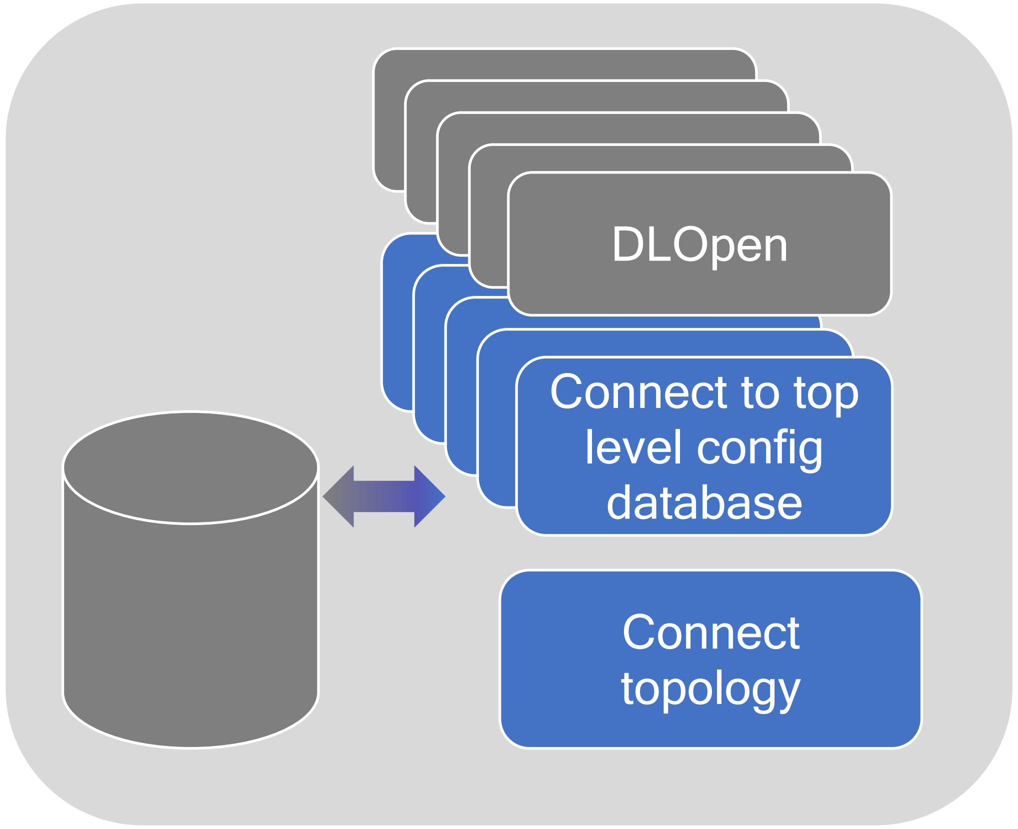
Nodes

- Nodes may implement any/all of the interfaces.
- e.g. A Configurator implements the 'config' interface, and uses the 'topology' interfaces.

- Each node is connected to other nodes for Configuration, Data, etc.
- Each node may expose many (hierarchical) interfaces that can be "bound"



Top level configurator scheme



The nature of the configuration database is up to the configurator.

Configurator must provide the standard interface to the database and provide that to the components

The configurator can then use the topology interface to connect nodes.

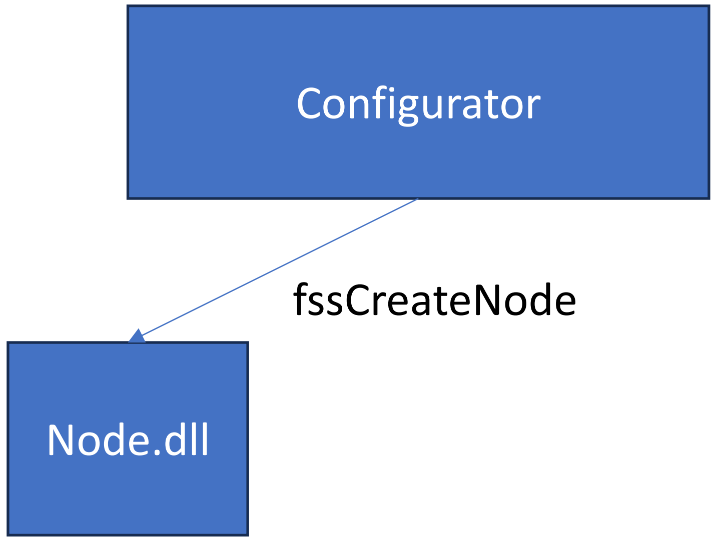
Of course there may be other databases, and other configurators within the system.

Morphology of the interface

Unashamedly borrowed from FMI/FMU

In the beginning

```
fssExport fssBindIFHandle
fssCreateNode(fssString nodeName,
fssConfigIfHandle configIfHandle);
```



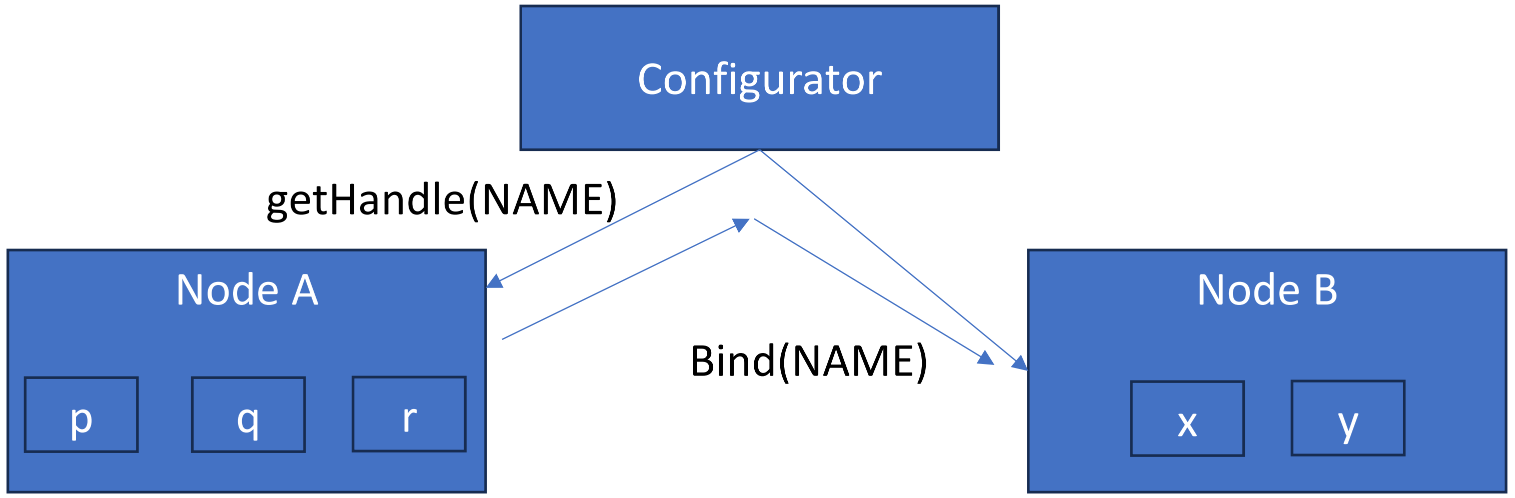
Provides a (unique) name
Provides a configuration API
Gets back a ‘Bind interface’

The Bind interface

```
typedef struct fssBindIf {
    fssUint64 version;
    fssBindEventsIfFnPtr bindEventsIf;
    fssBindBusIfFnPtr bindBusIf;
    fssBindTimeSyncIfFnPtr bindTimeSyncIf;
    fssBindControlIfFnPtr bindControlIf;
    fssGetEventsIfHandleFnPtr getEventsIfHandle;
    fssGetBusIfFnPtr getBusIfHandle;
    fssGetTimeSyncIfHandleFnPtr getTimeSyncIfHandle;
    fssGetControlIfHandleFnPtr getControlIfHandle;
} fssBindIf;
```

- Basic structure exported by all nodes
- Nodes need not implement all functions.

Get and Bind



```
h=getHandle("A.q")
Bind(h,"B.y")
```

Step 1: Request a handle from one node.
Step 2: Bind that handle to another node.

Both identified by a NAMES

NB, to keep things simple, interfaces are unidirectional
B knows about A, but A may not know it has been bound to B

Simulation Event Interface

```
typedef void (*fssEventHandlerFnPtr)(fssEventsIfHandle eventsIfHandle, fssUint64 event);
```

- NOTE: As with all interfaces, once established, this simply provides a means by which one node can tell another node about an ‘event’

NOTHING MORE

- What does that event “mean” – Go to the Git repo!
- You want to send to all - you need a node to broadcast!

Time Sync: From Time To Time

- Between any two simulators, simply exchange time windows

  {From Time, To Time}

- Always try to advance to the “from” time,
- Try not to advance beyond the “to” time
- When you arrive at the “to” time (or at any point), send a nev window to the other side.

- Can be used to handle FMI/FMU, Silkit, SystemC, EDA247, etc

Sync algorithms : Just how you connect!

Steppers

Central controller

"Do step" – and wait for everybody to complete

Watchers

Central time source

Everybody should do their best to be in sync

Talkers

Each node broadcasts it's time (window) along with all communication

Each node should do their best to stay in sync with it's neighbours

Implementing nodes as "talkers" allows them to operate within a central controller (either time, or stepper based)

Simulation Event Interface

```
typedef void (*fssEventHandlerFnPtr)(fssEventsIfHandle eventsIfHandle, fssUint64 event);
```

- NOTE: As with all interfaces, once established, this simply provides a means by which one node can tell another node about an 'event'

NOTHING MORE

- What does that event "mean" – Go to the Git repo!
- You want to send to all - you need a node to broadcast!

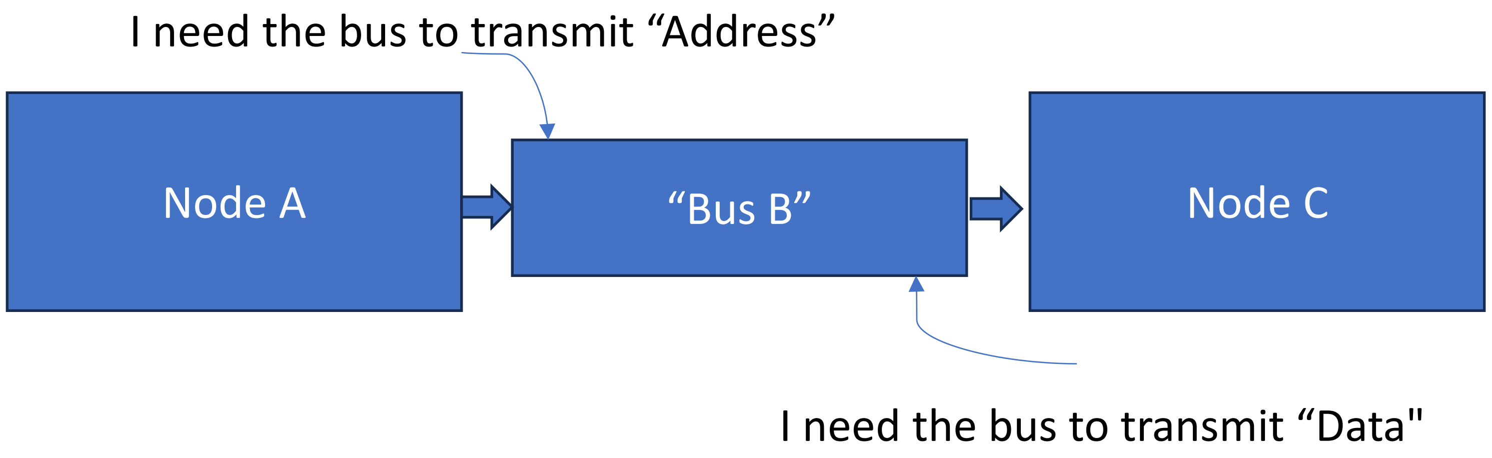
Simulation Event Interface

```
typedef fssString (*fssDoCommandFnPtr)(fssControlIfHanlde controlIfHandle,
fssString cmd);
```

- Simple “JSON” string
- For definitions – use Git Repo

Data Interface? – More a sort of “meta bus”

Think: “Dynamic Protocol Buffers”


I need the bus to transmit “Address”
Node A
“Bus B”
Node C
I need the bus to transmit “Data"

Each node lists the items that it needs on the bus
Names (and semantics) : On the Git repo (!)
Then simple API to set/get items from the “bus” which organizes the transmission.

`bus’ container

- The abstraction of a ‘bus’ is basically a dynamic protobuf container.
- The interface is used to:
  - Describe the ‘bus’ container.
  - Add/extract data from the container.
  - Indicate that the container should be processed by other nodes that have access to it.
    - This last could be an ‘event’ but it’s convenient to keep the API together
    - Open question : Should we add time, by default, to this event?

Simulation Event Interface

```
typedef struct fssBusIf {
    fssUint64 version;
    fssBusIfGetItemsNumberFnPtr getNumber;
    fssBusIfGetNameFromIndexFnPtr getName;
    fssBusIfGetSizeFnPtr getSize;
    fssBusIfGetIndexFnPtr getIndex;
    fssBusIfAddItemFnPtr addItem;
    fssBusIfGetItemFnPtr getItem;
    fssBusIfSetItemFnPtr setItem;
    fssBusIfTransmitFnPtr transmit;
} fssBusIf;
```

Pub Sub?

- Each API can be wrapped into a Pub/Sub interface
- Prof of concept with Zenoh
- Each name is unique in the design

- Since the name binding is just a string – wildcards can be made to work.

POC?

- Using Zenoh and/or SystemC
- Available as part of qbox

https://github.com/quic/qbox