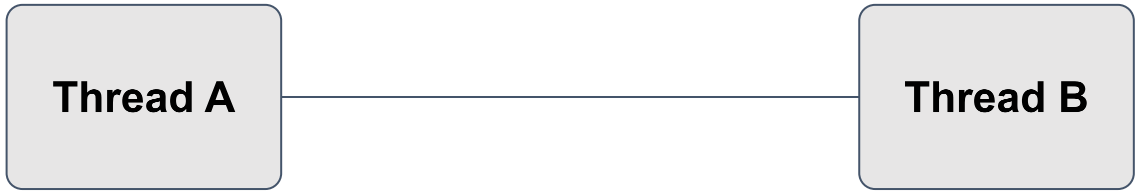
Zephyr® Project
Developer Summit

ZBus - the lightweight and flexible Zephyr message bus

Rodrigo Peixoto, *Edge-UFAL/Citrinio*
@rodrigopex

Motivation

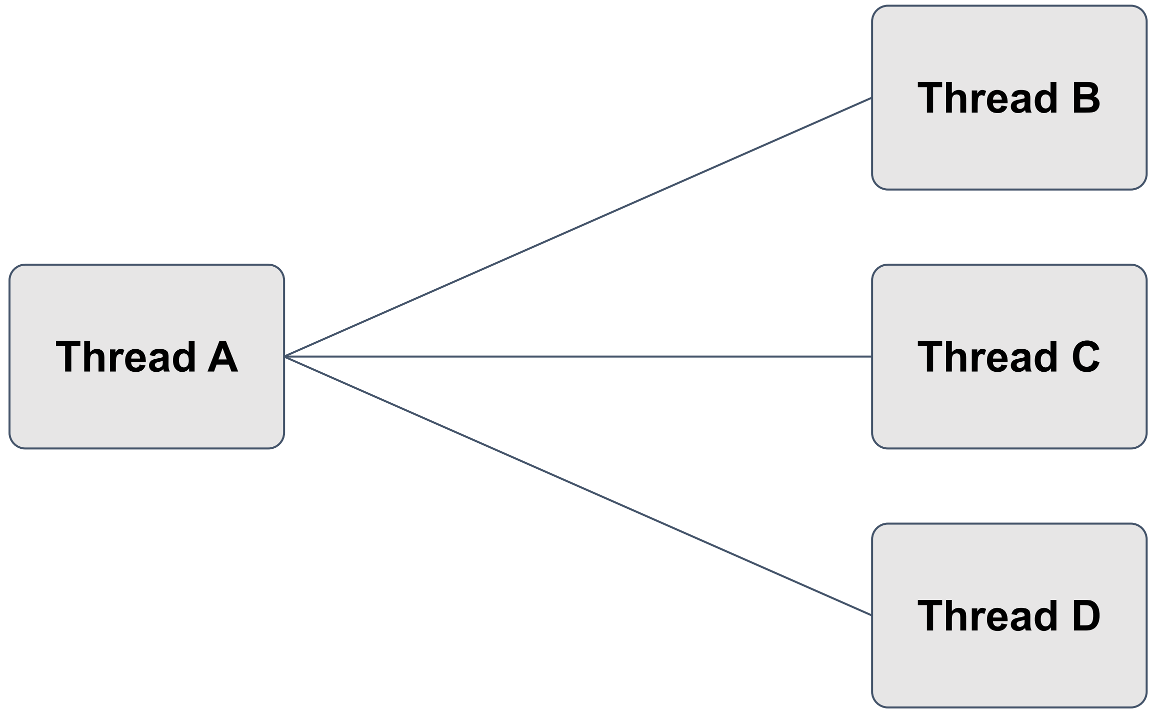
ONE-TO-ONE



FIFO	✅
LIFO	✅
Stack	✅
Message queue	✅
Mailbox	✅
Pipe	✅

Motivation

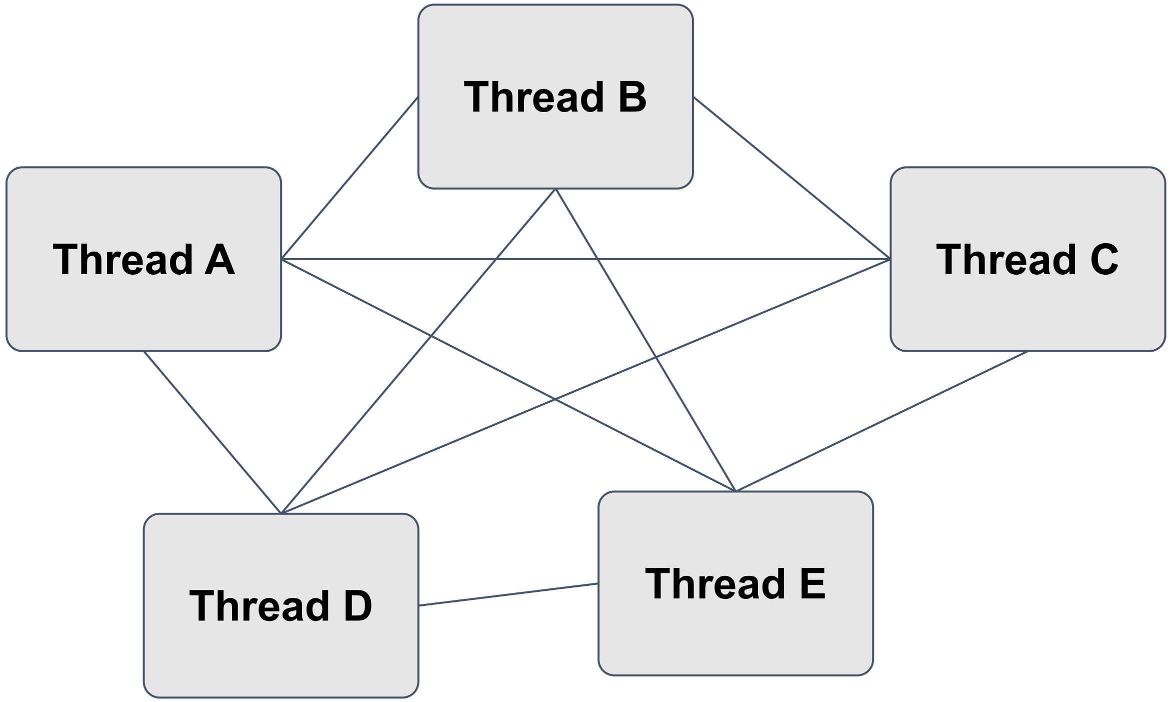
ONE-TO-MANY



FIFO	❌
LIFO	❌
Stack	❌
Message queue	❌
Mailbox	—
Pipe	❌

Motivation

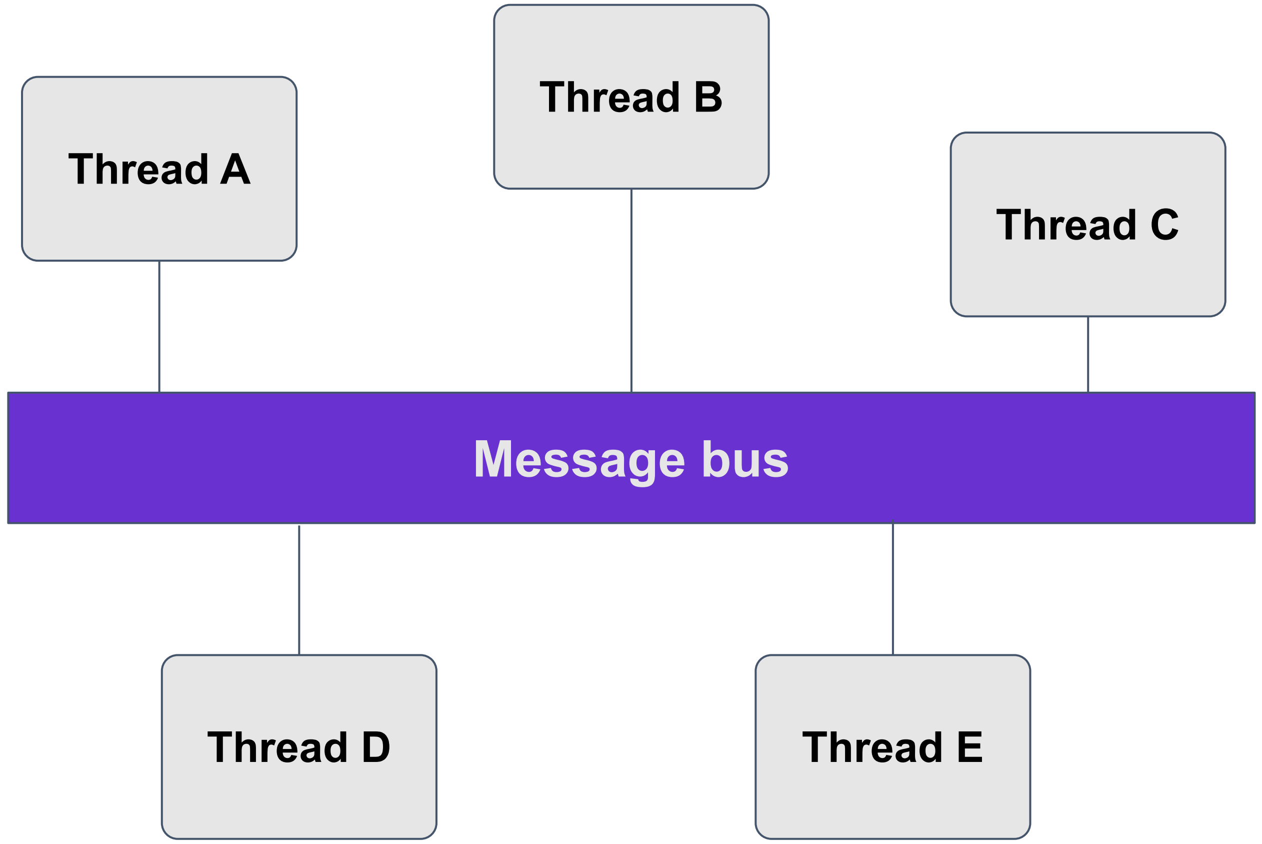
MANY-TO-MANY



FIFO	❌
LIFO	❌
Stack	❌
Message queue	❌
Mailbox	➖
Pipe	❌

Zephyr® Project
Developer Summit

Solution idea

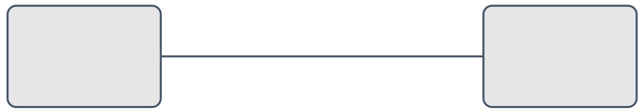


FIFO	❌
LIFO	❌
Stack	❌
Message queue	❌
Mailbox	❌
Pipe	❌
Bus	✅

Bus topologies

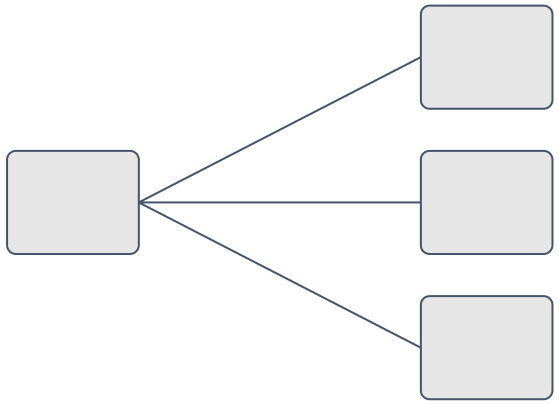
✅

ONE-TO-ONE



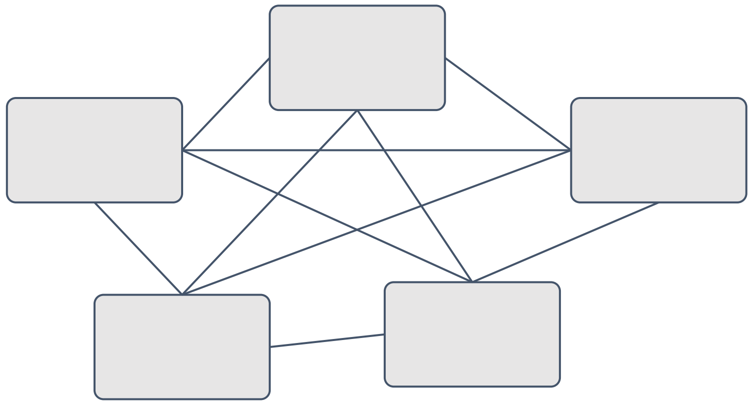
✅

ONE-TO-MANY



✅

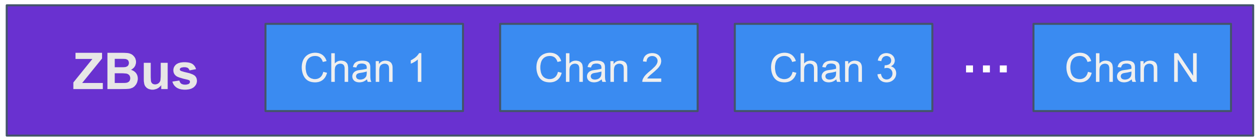
MANY-TO-MANY



Embedded systems challenges

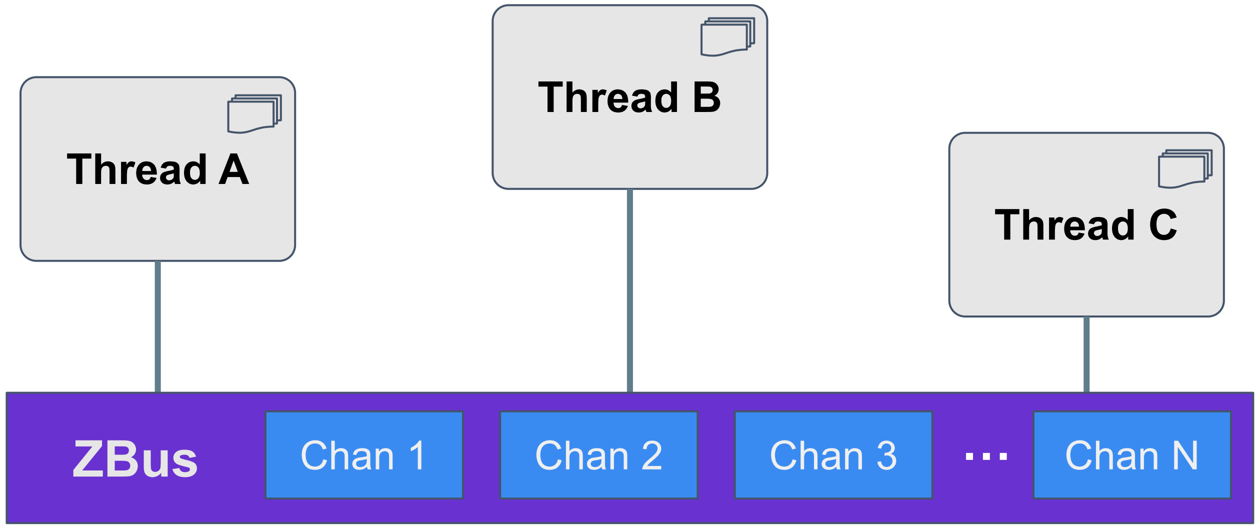
- Memory constraints
- Processing limitations
- Battery-powered devices

Zephyr® Project
Developer Summit


ZBus
Chan 1
Chan 2
Chan 3
...
Chan N

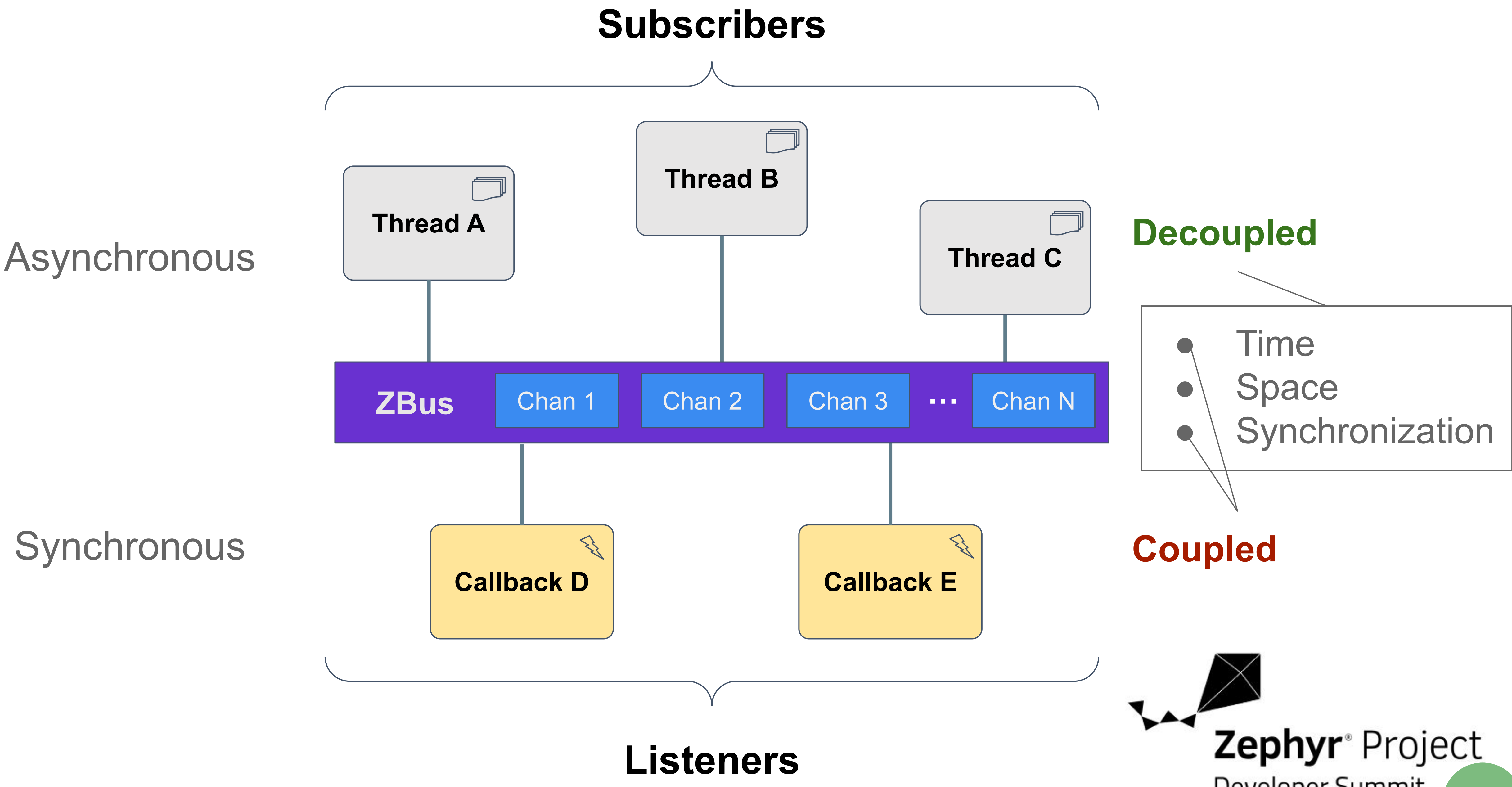
Subscribers

Asynchronous

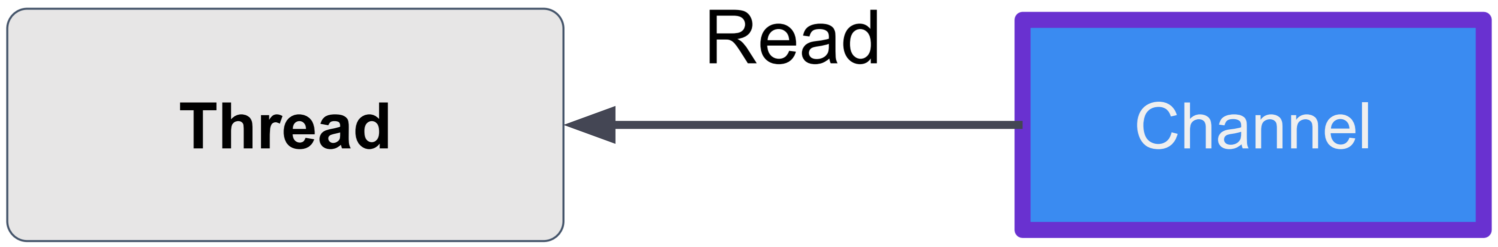


Decoupled

- Time
- Space
- Synchronization

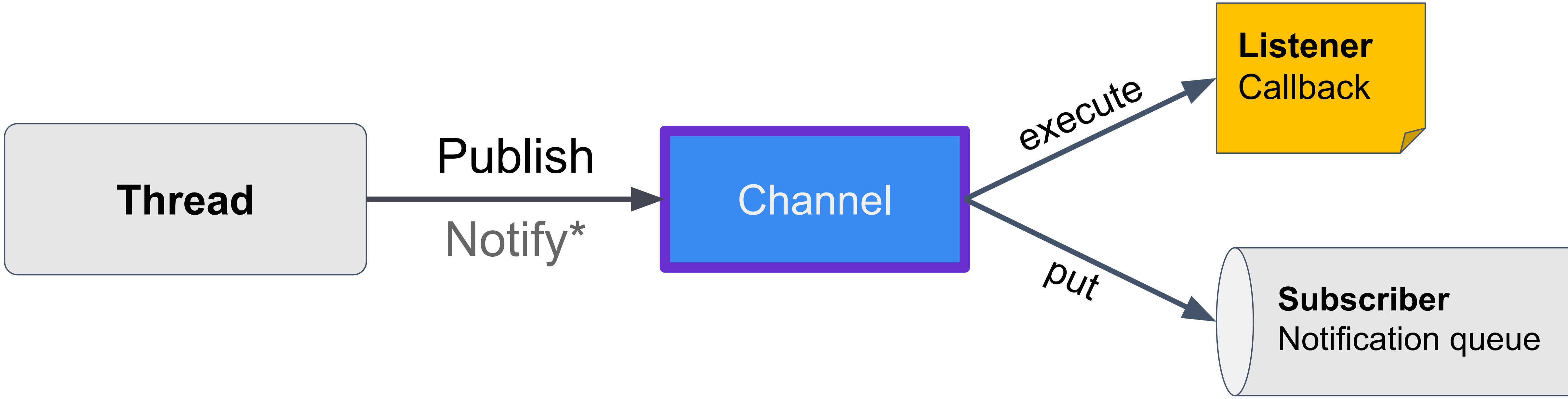

Subscribers
Asynchronous
Thread A
Thread B
Thread C
ZBus
Chan 1
Chan 2
Chan 3
...
Chan N
Synchronous
Callback D
Callback E
Listeners
Decoupled
Time
Space
Synchronization
Coupled

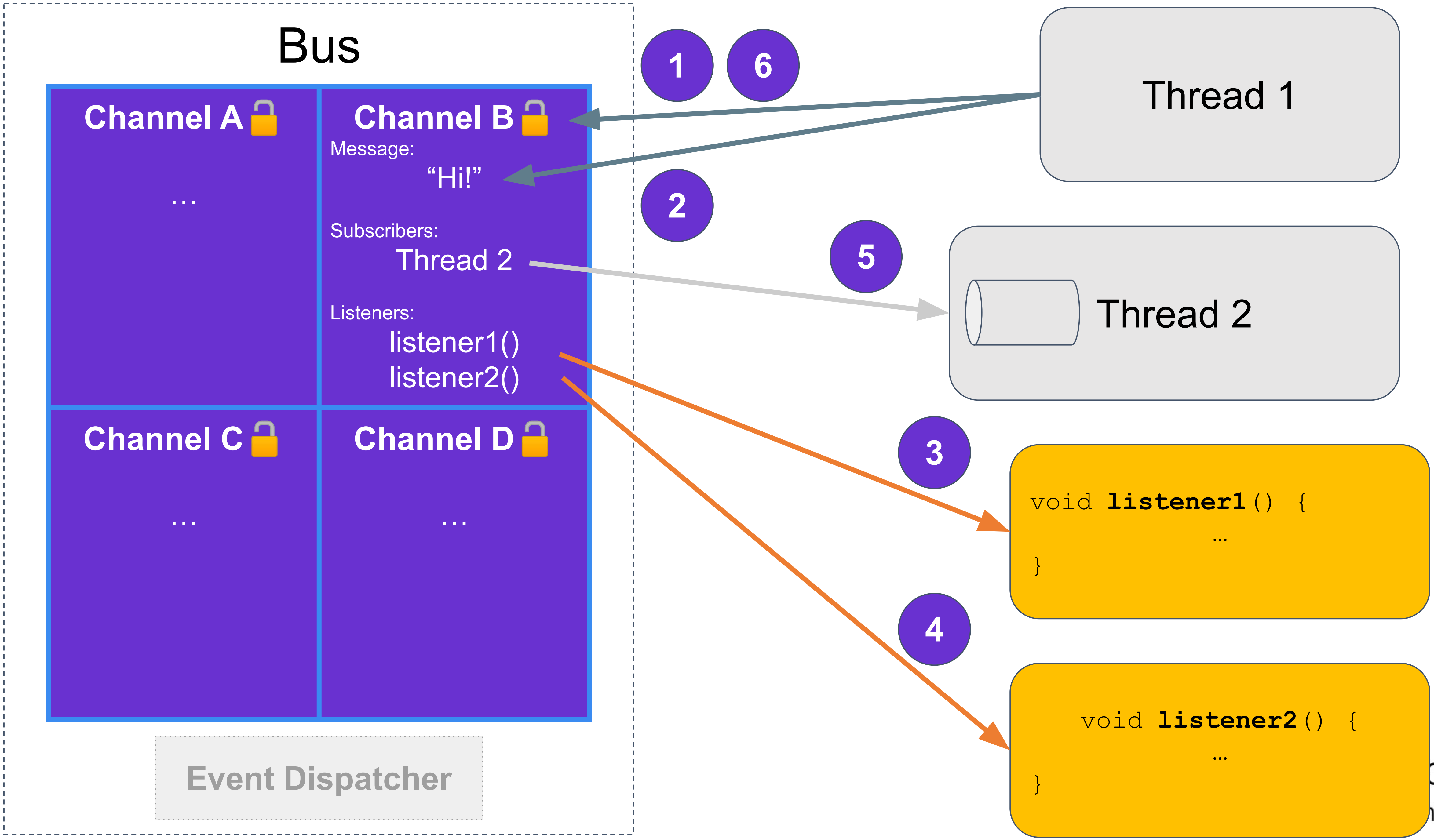
Available actions



Zephyr® Project
Developer Summit

Available actions





```
void listener1() {
        …
}
```

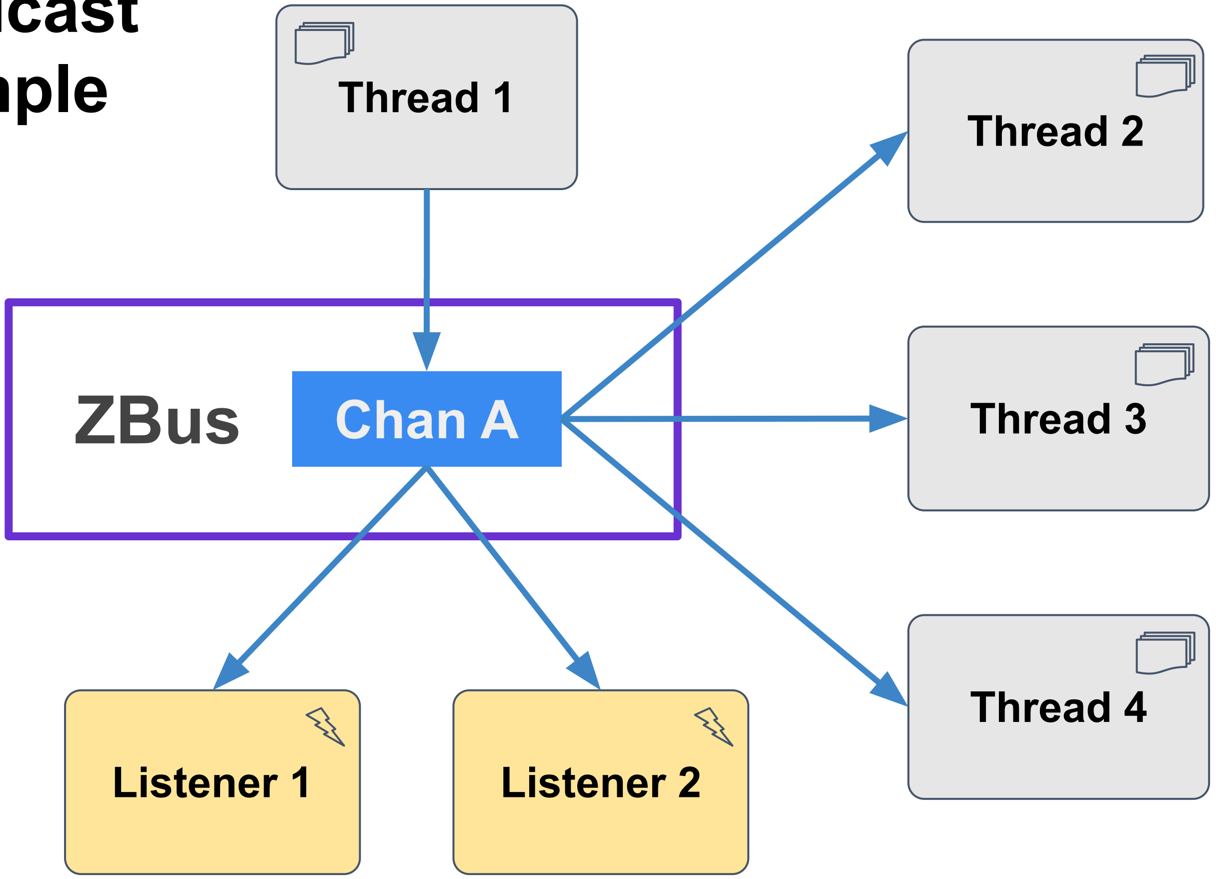
```
void listener2() {
        …
}
```

Virtual Distributed Event Dispatcher

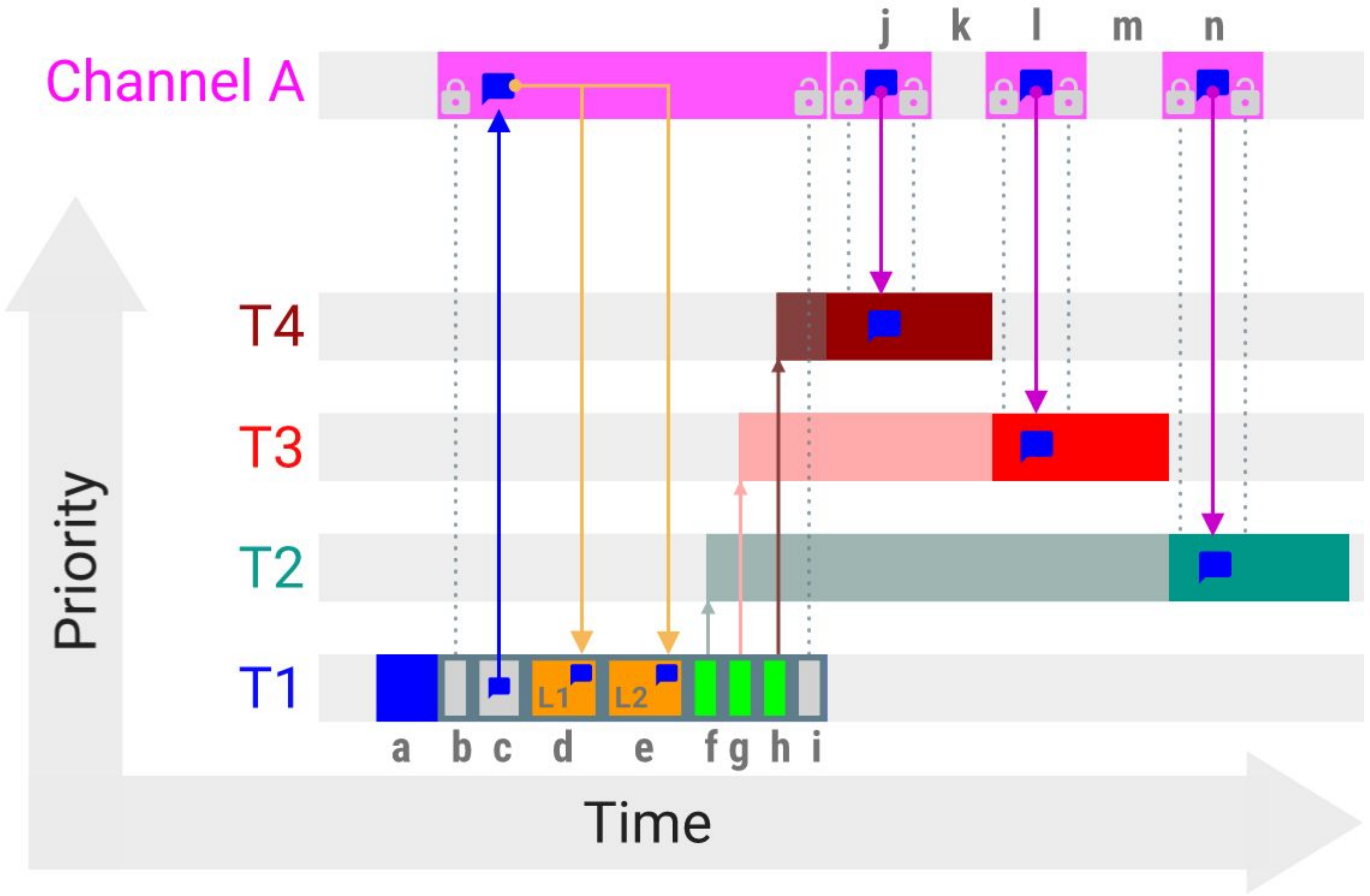
VDED is the bus logic responsible for sending notifications about message publications to the channel's observers. There is no central entity that acts as an event dispatcher on ZBus.

Virtual Distributed Event Dispatcher

Broadcast example

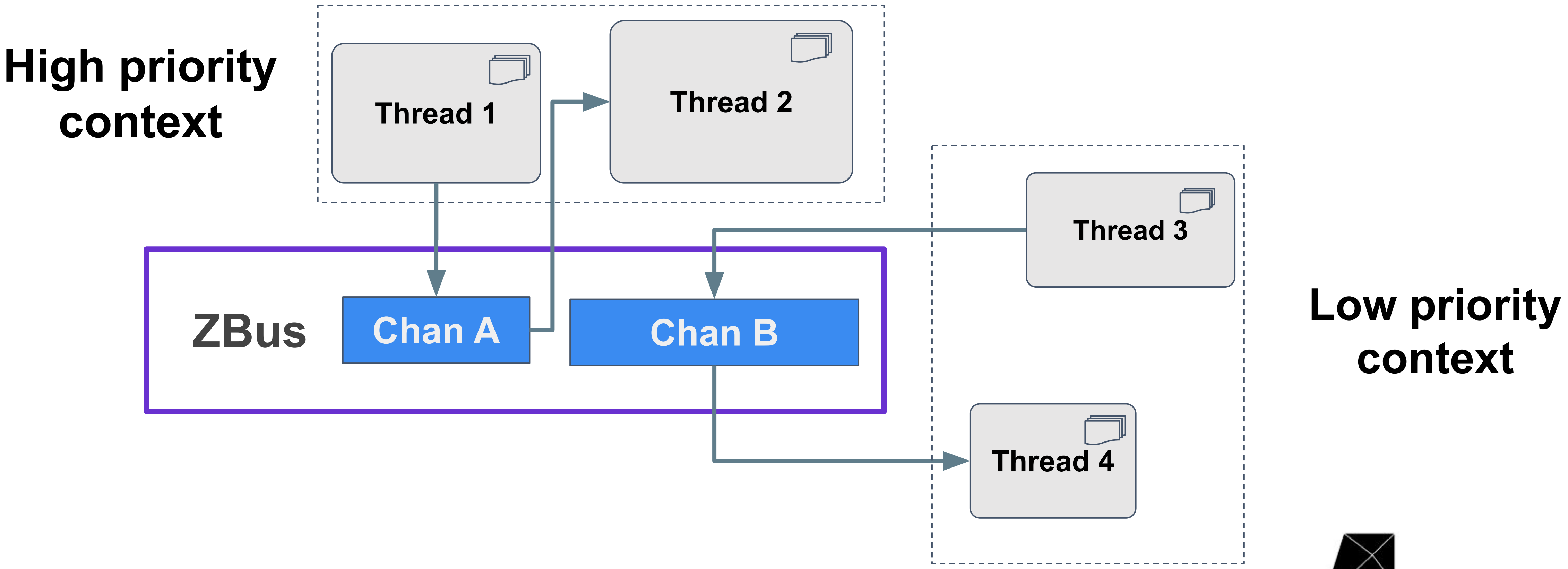


Virtual Distributed Event Dispatcher

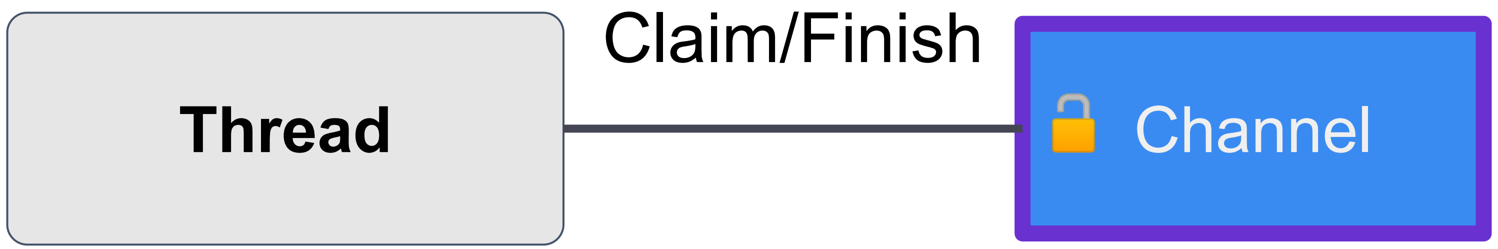


Zephyr® Project
Developer Summit

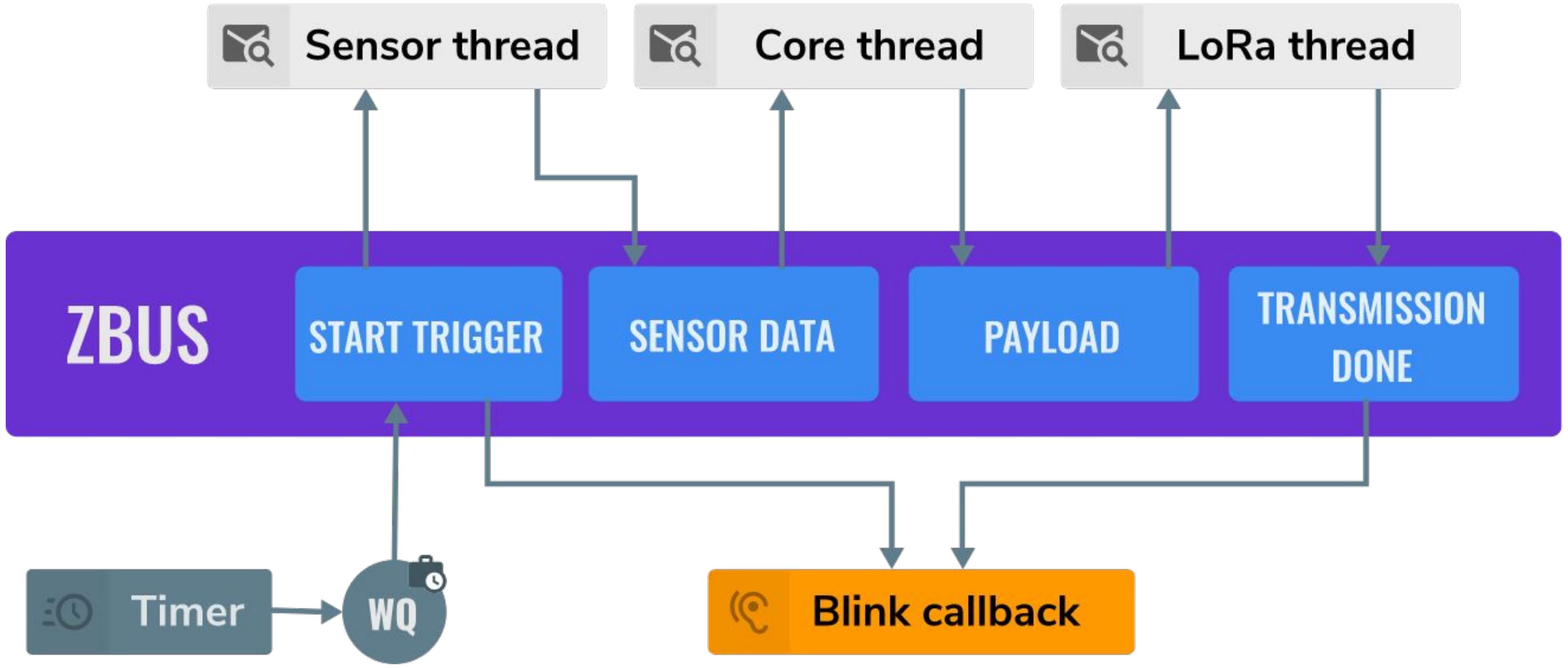
Virtual Distributed Event Dispatcher



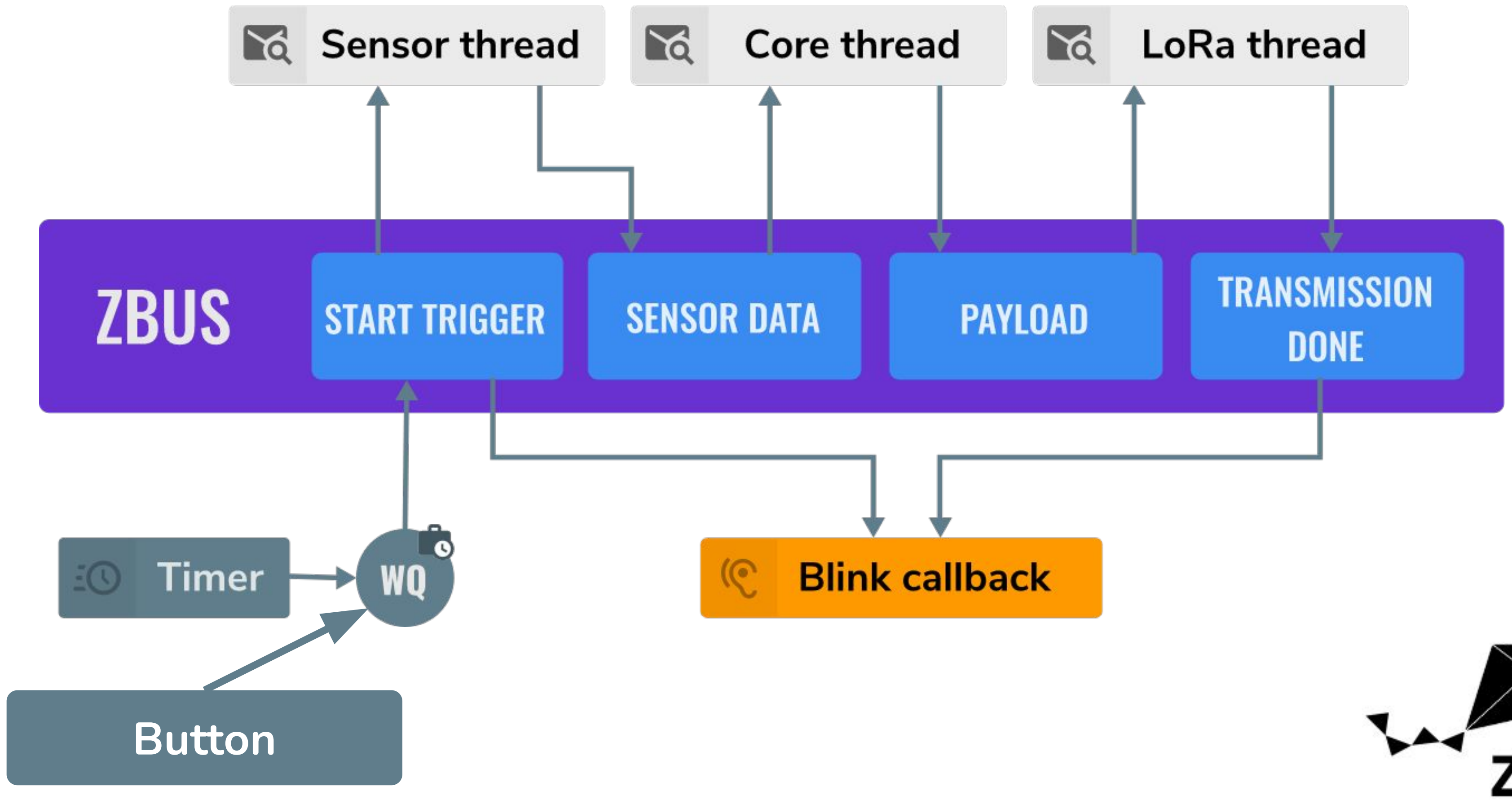
Available actions



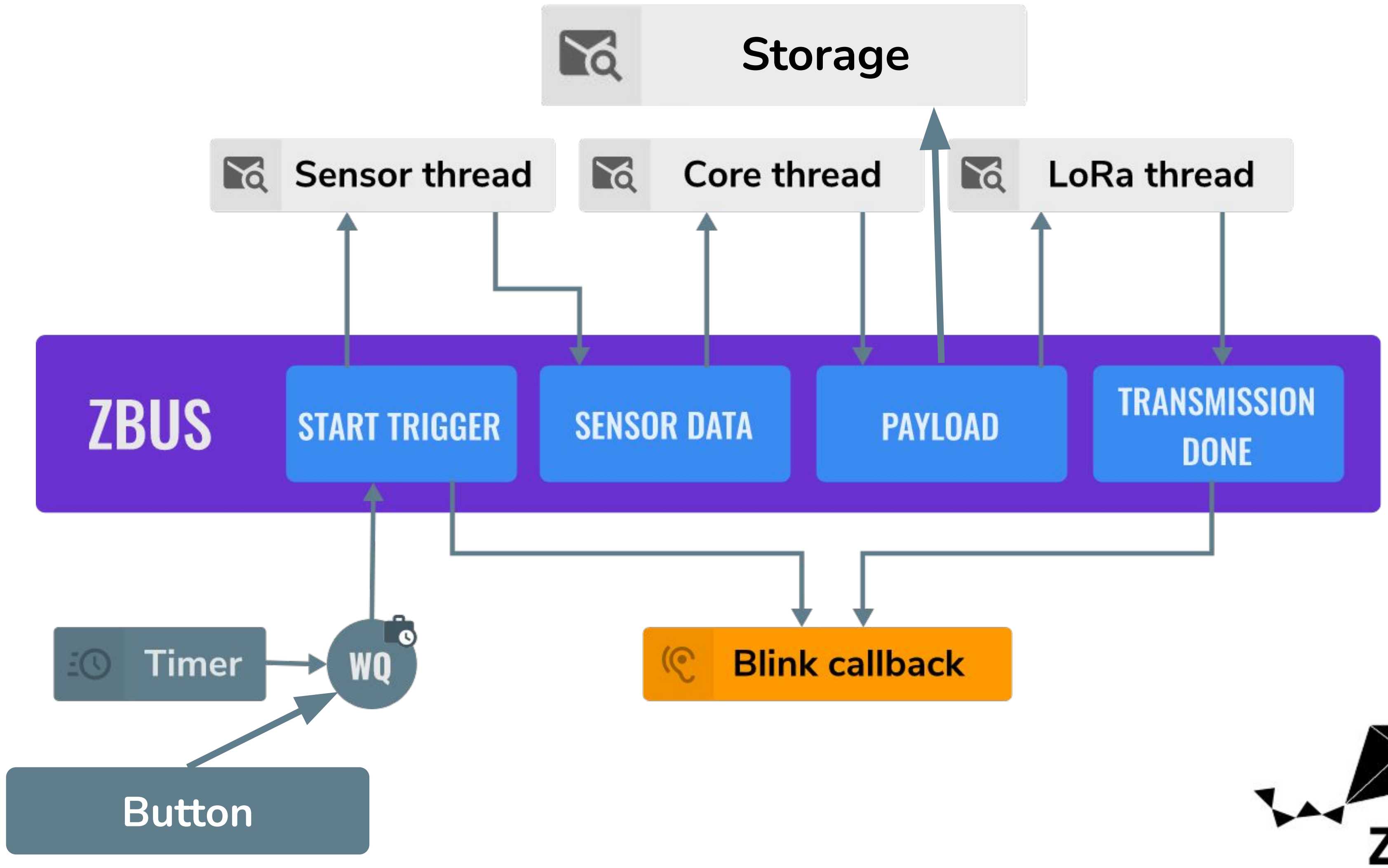
Example


Sensor thread
Core thread
LoRa thread
ZBUS
START TRIGGER
SENSOR DATA
PAYLOAD
TRANSMISSION DONE
Timer
WQ
Blink callback

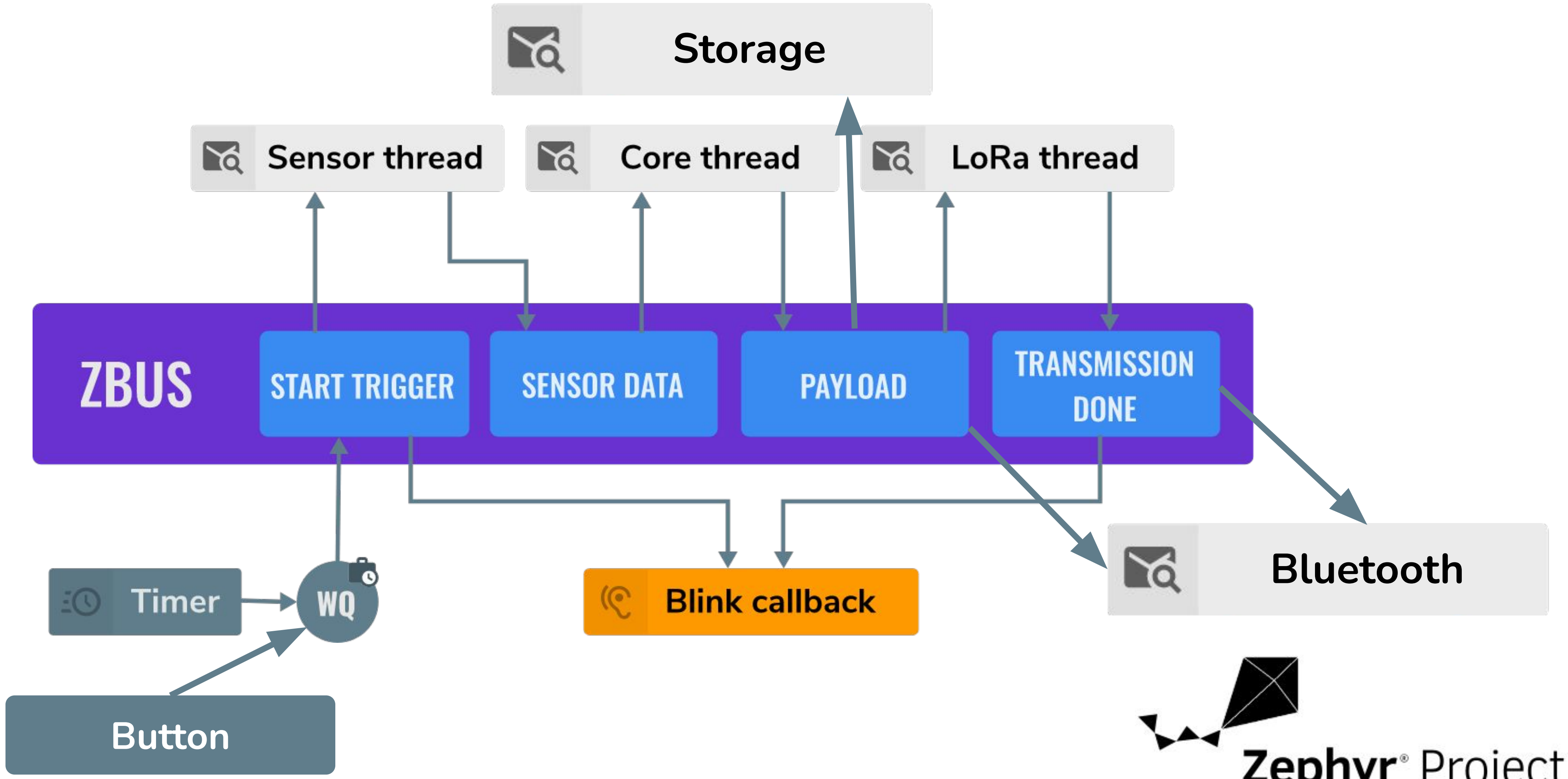
Example


Sensor thread
Core thread
LoRa thread
ZBUS
START TRIGGER
SENSOR DATA
PAYLOAD
TRANSMISSION DONE
Timer
WQ
Blink callback
Button

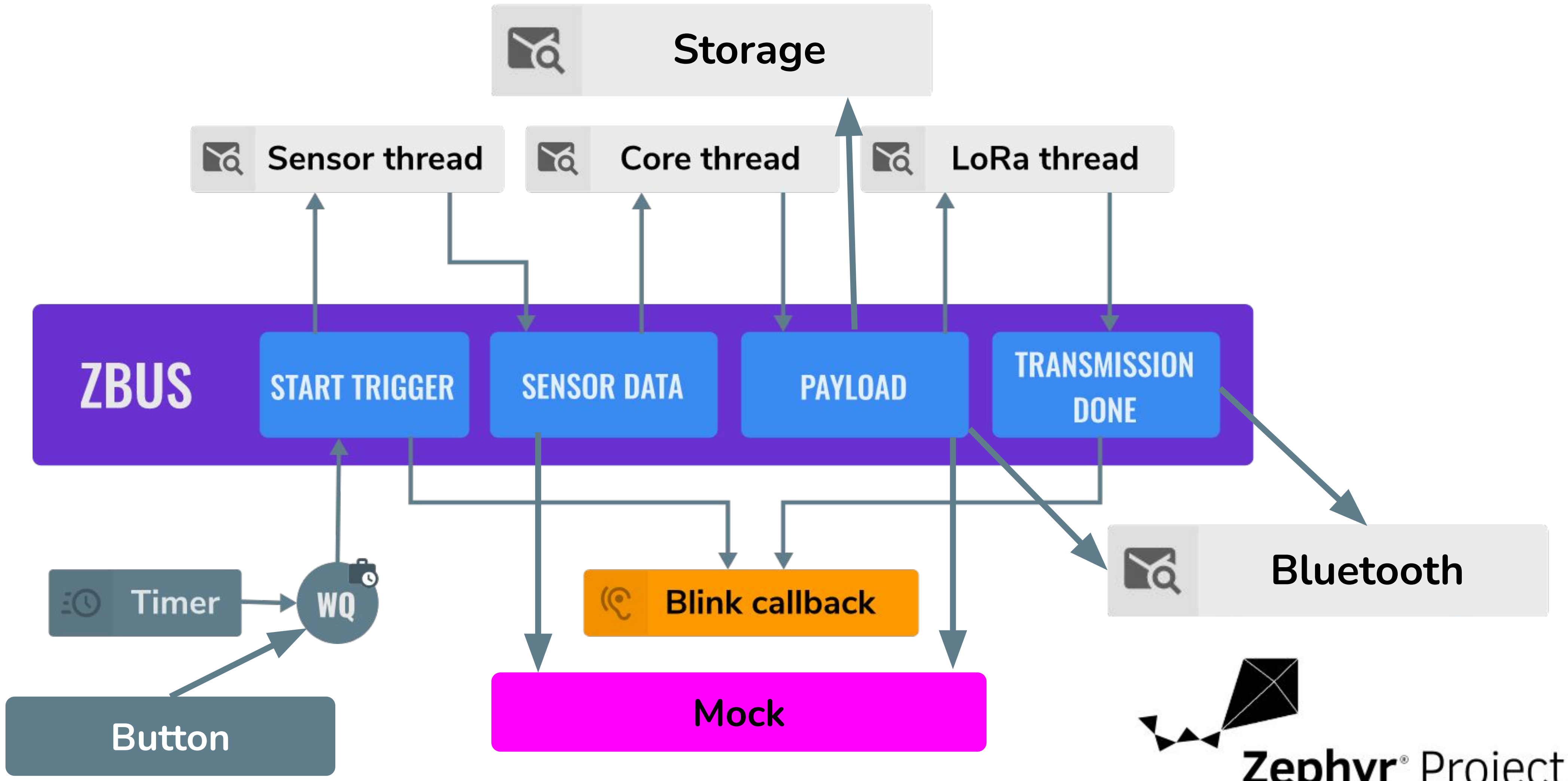
Example


Storage
Sensor thread
Core thread
LoRa thread
ZBUS
START TRIGGER
SENSOR DATA
PAYLOAD
TRANSMISSION DONE
Timer
WQ
Blink callback
Button

Example

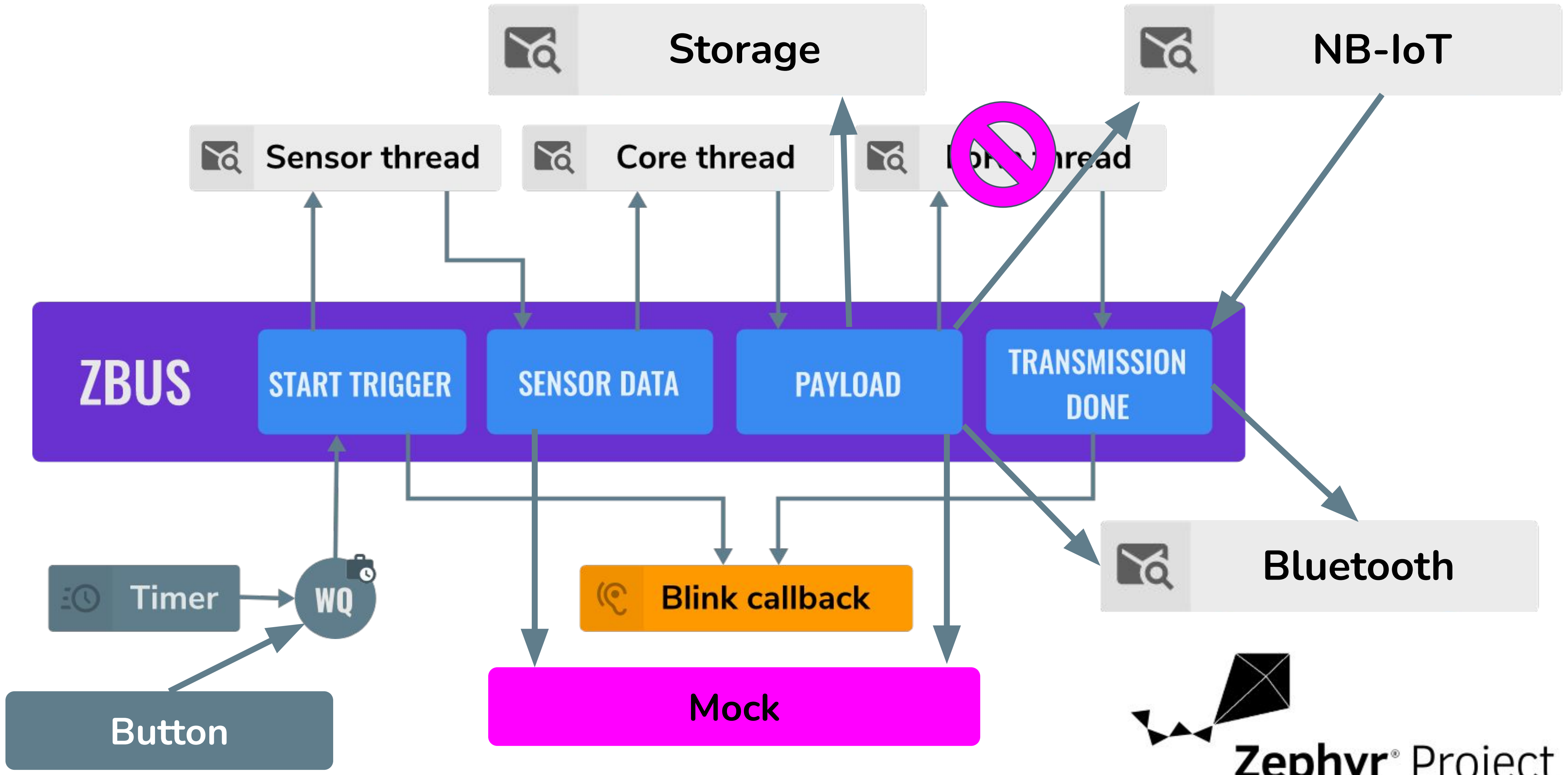

Storage
Sensor thread
Core thread
LoRa thread
ZBUS
START TRIGGER
SENSOR DATA
PAYLOAD
TRANSMISSION DONE
Timer
WQ
Button
Blink callback
Bluetooth

Example



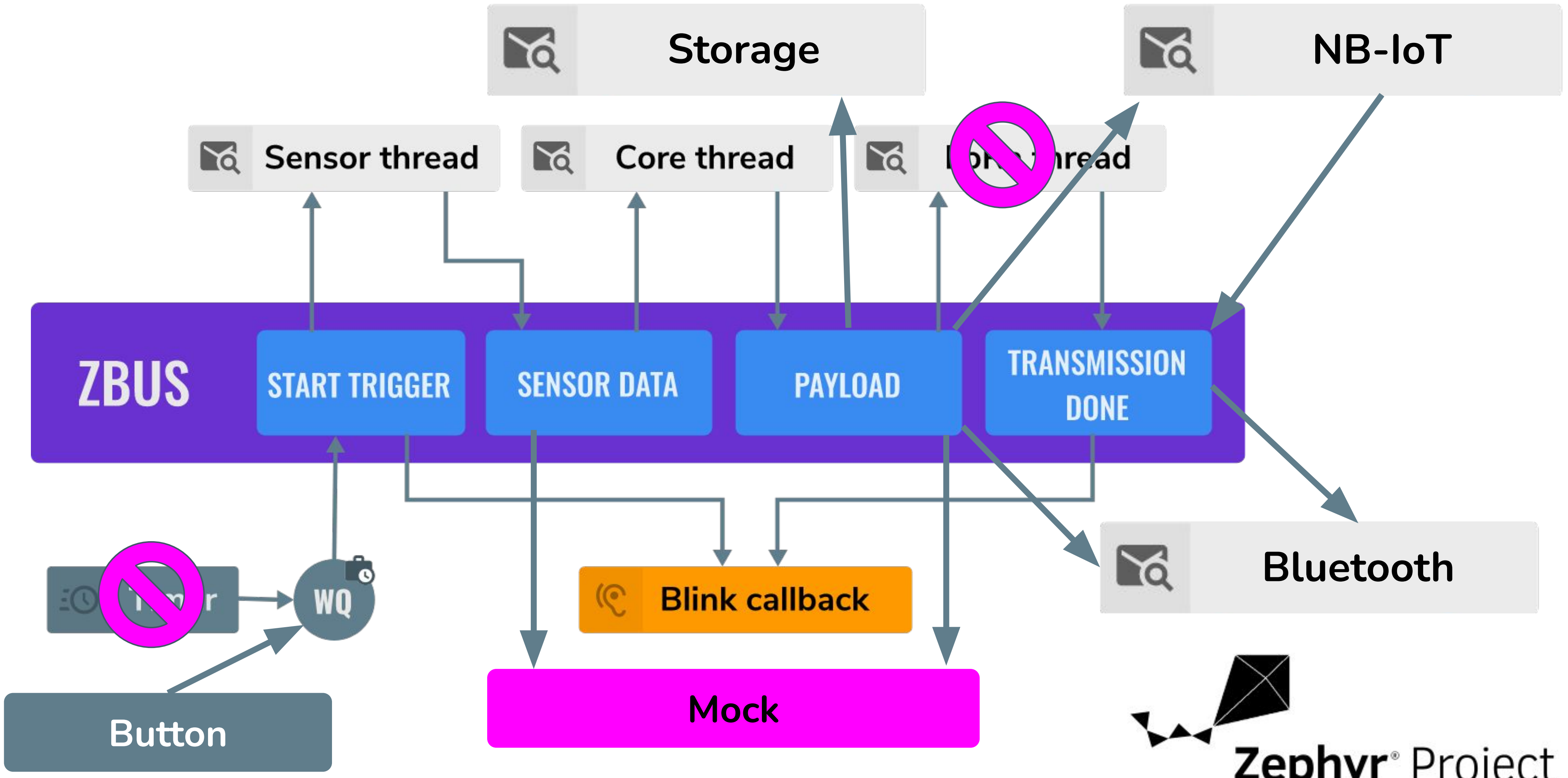
Zephyr® Project
Developer Summit

Example



Zephyr® Project
Developer Summit

Example



Zephyr® Project
Developer Summit

Usage considerations

PROS

- Promotes event-driven architecture
- Unified way to make threads talk and share data
- Code decoupling (time, space, and synchronization)*
- Promotes reuse
- Increase testability (+controllability +observability)
- Extensible (claim/finish + user_data)

CONS

- Take time to master it
    - Too many possibilities
- Not for intensive byte streaming
- No delivery guarantees for subscribers

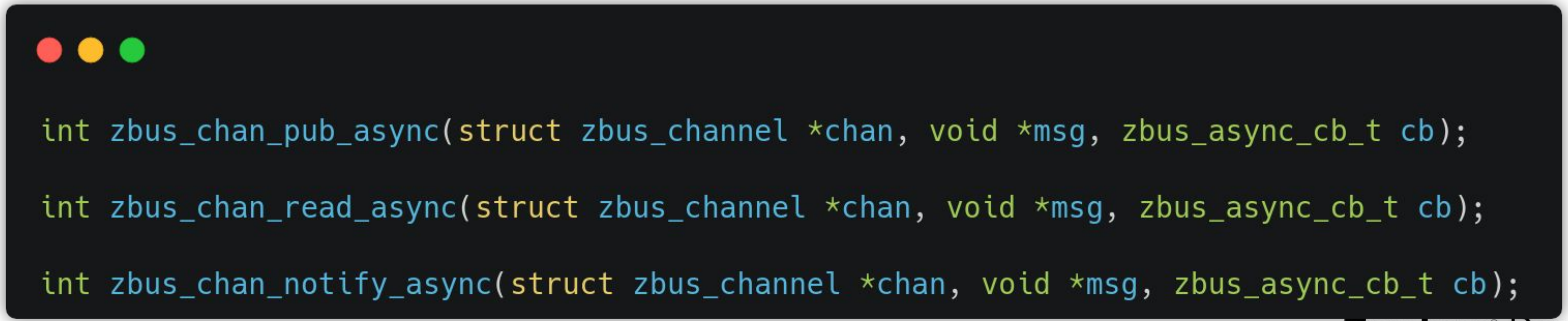
Zephyr® Project
Developer Summit

Zephyr® Project
Developer Summit

ZBus Features Backlog

ZBus async APIs

- Run inside an ISR could be possible
- Would avoid using work queues
- Dedicated ZBus thread



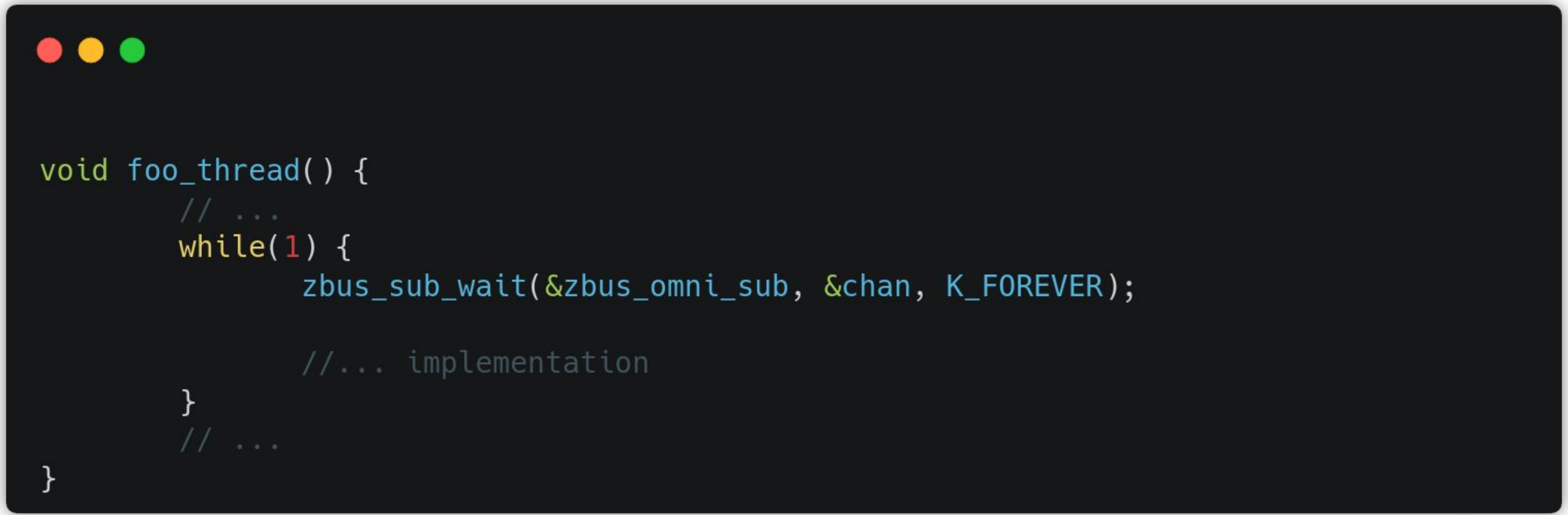
```c
int zbus_chan_pub_async(struct zbus_channel *chan, void *msg, zbus_async_cb_t cb);

int zbus_chan_read_async(struct zbus_channel *chan, void *msg, zbus_async_cb_t cb);

int zbus_chan_notify_async(struct zbus_channel *chan, void *msg, zbus_async_cb_t cb);
```

ZBus omni subscriber

- The omni subscriber will listen to all the channels
- It can be used to extend the bus features



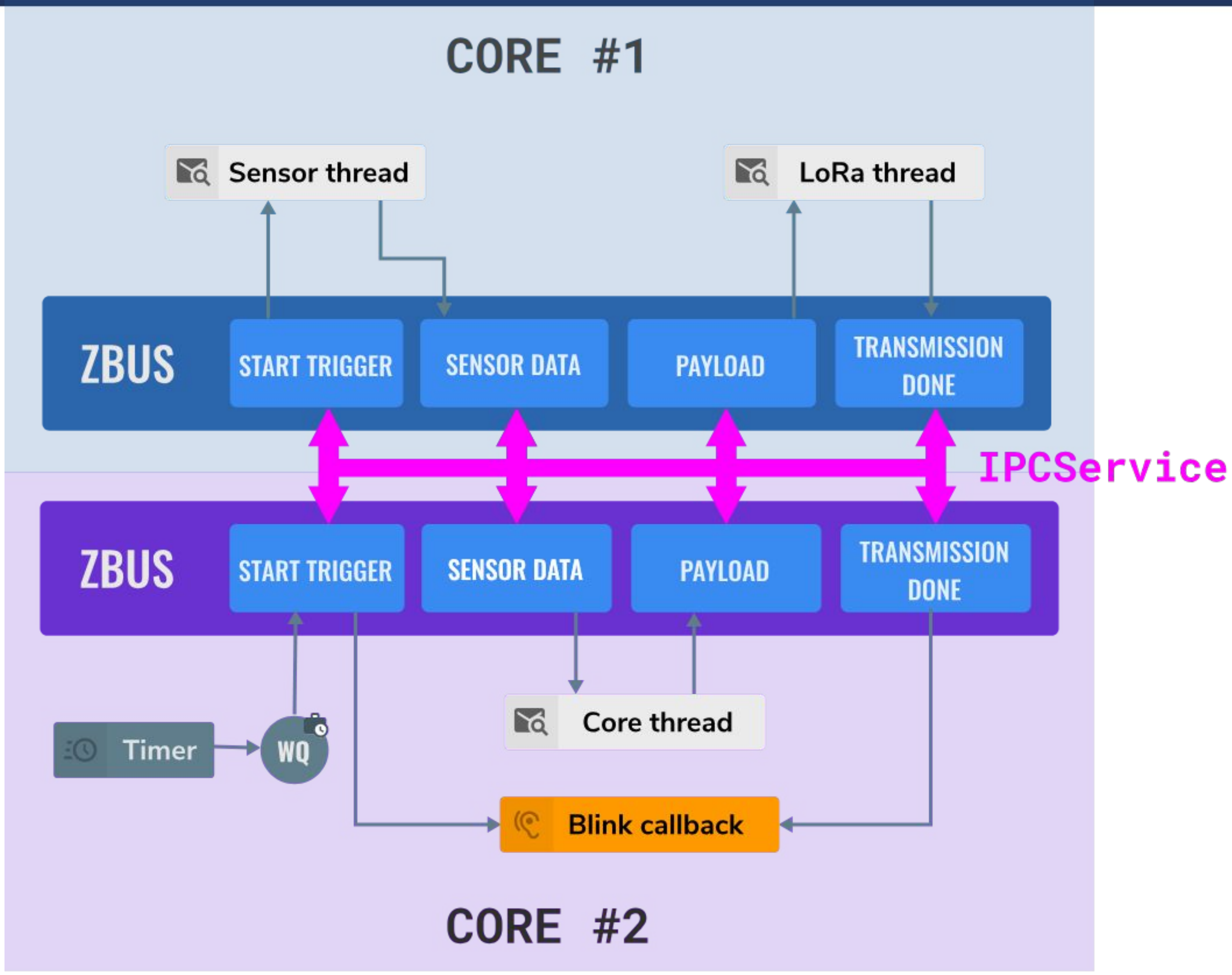
```
void foo_thread() {
        // ...
        while(1) {
                zbus_sub_wait(&zbus_omni_sub, &chan, K_FOREVER);

                //... implementation
        }
        // ...
}
```

Integration and samples

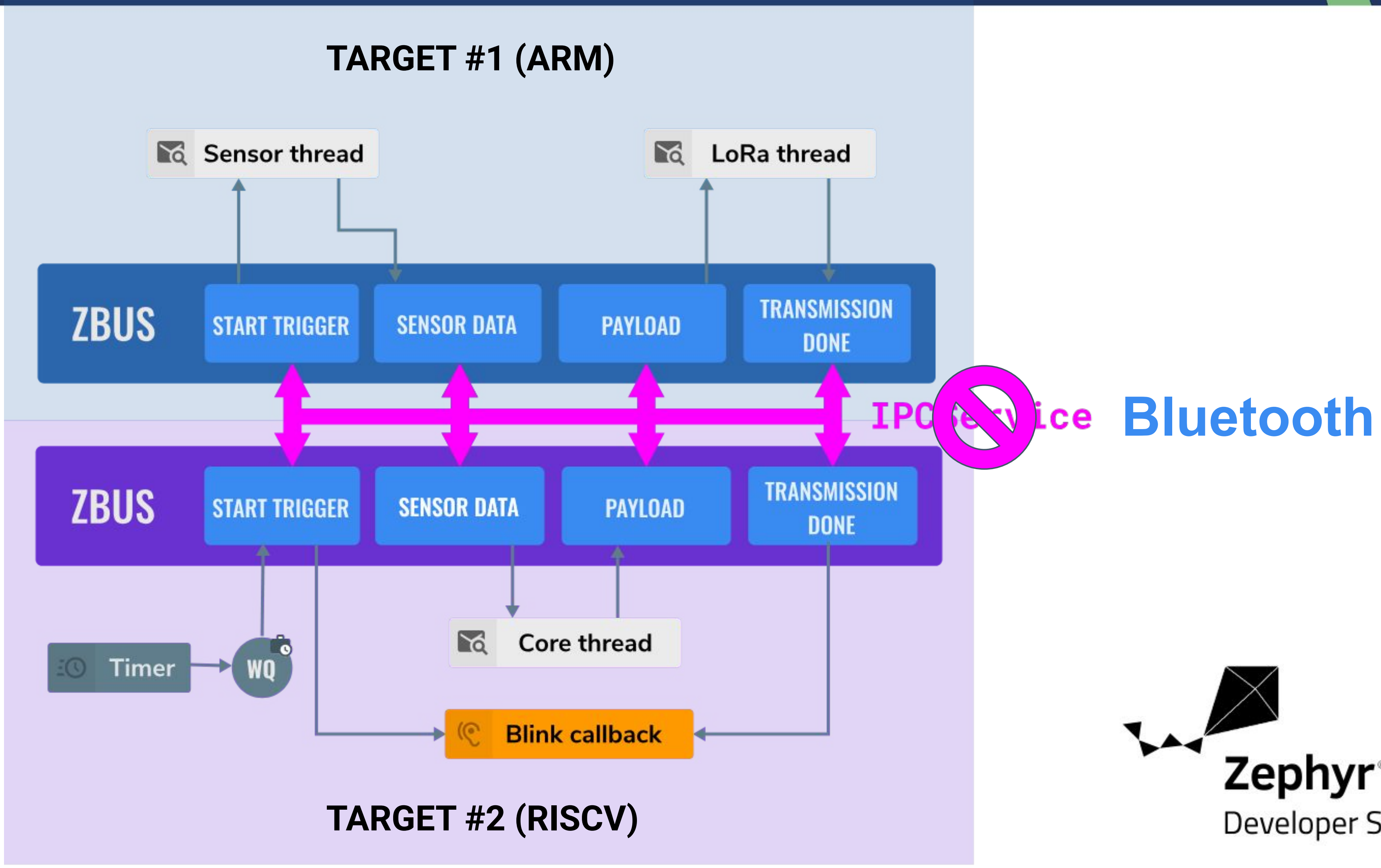
- Make ZBus an Input subsys event distributor backend
- Add samples (Bluetooth, Sensors, FSM, etc.)

ZBus for multi-core

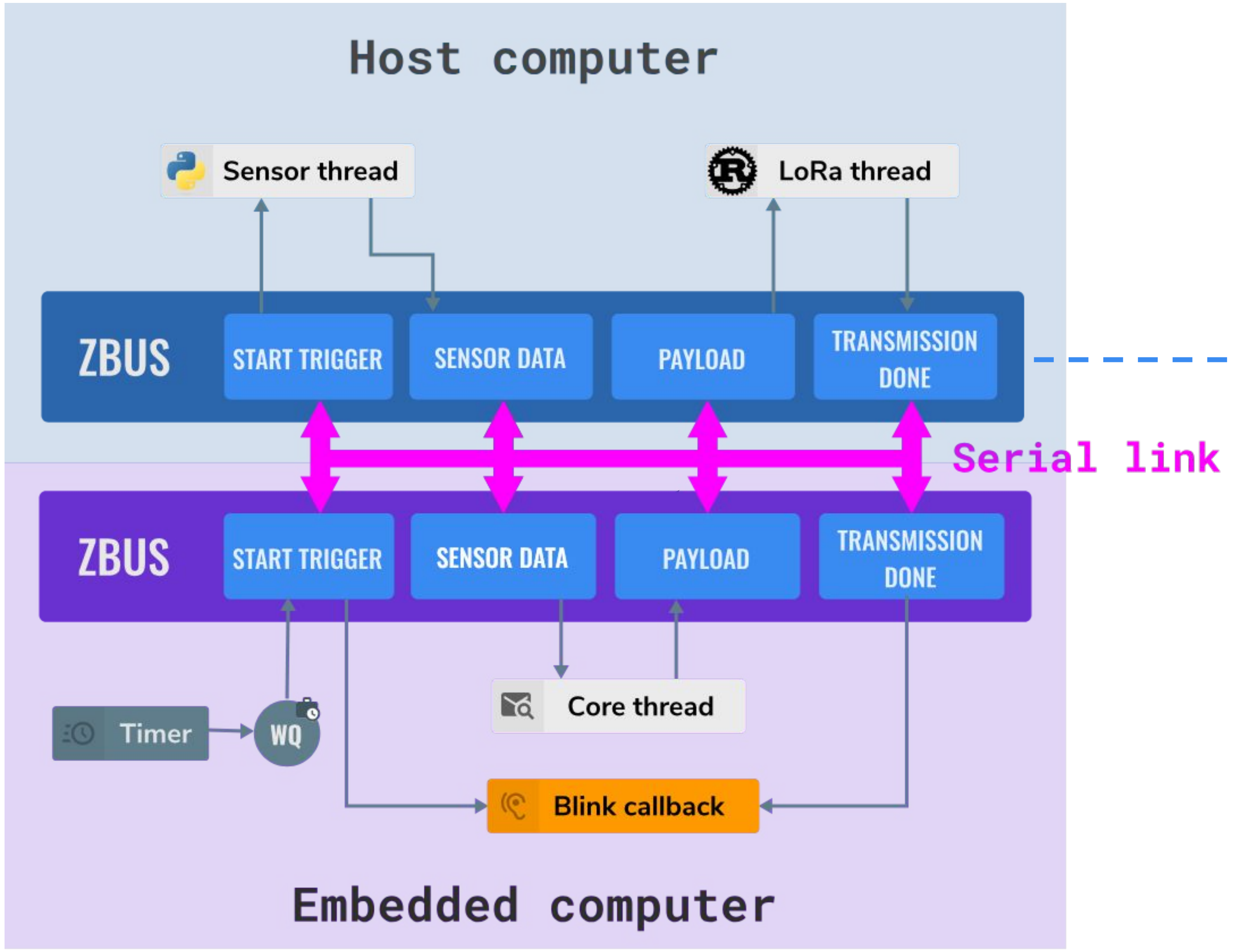


Zephyr® Project
Developer Summit

ZBus for multi-target



ZBus desktop



+

Zephyr® Project
Developer Summit

ZBus Discord Channel

ZBus Roadmap topic at Zephyr Discord channel

Zephyr® Project
Developer Summit

Tips and tricks

Listeners

❌ Avoid excessive use of them, they are running during the publishing process

❌ Do not sleep inside listeners. It will increase the publishing latency

✅ Think of them as an ISR. They must run as quickly as possible

Zephyr® Project Developer Summit

Listeners

✅ Use a work queue or separated thread instead of executing something heavy inside a listener

✅ Use **`zbus_chan_const_msg`** inside listeners. The channels are already locked!

Zephyr® Project
Developer Summit

Subscribers

❌ Do not use subscribers when losses and duplications cannot be tolerated

✅ Use listeners in conjunction with message queues

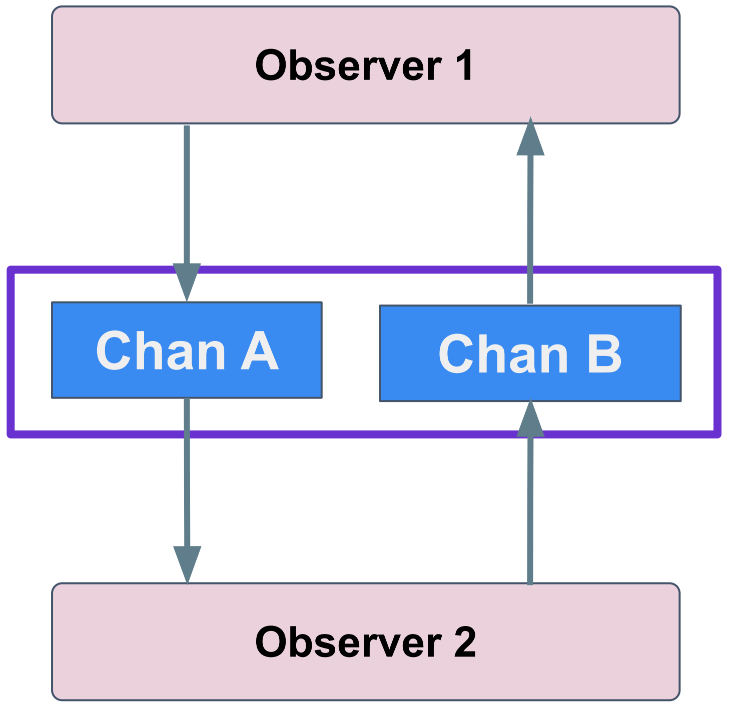
✅ PR with confirmed channels sample submitted

Zephyr® Project
Developer Summit

Undesired loops

❌ Take care with publishing loops

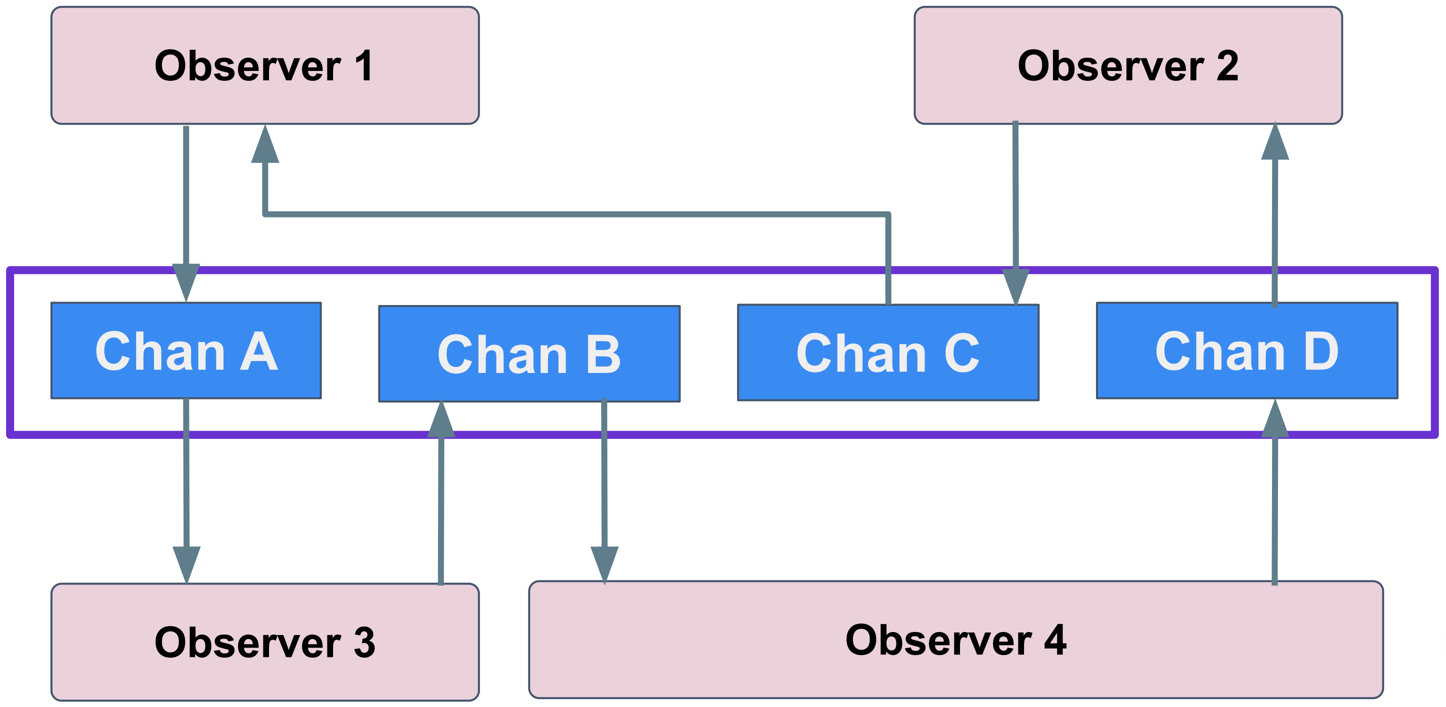
✅ Avoid loops on the bus diagram



Zephyr® Project
Developer Summit

Undesired loops

❌ Take care with chained publishing loops



ISR

❌ Do not use ZBus functions inside an ISR

✅ Postpone that by using work queues instead

Zephyr® Project
Developer Summit

Extras

- ✅ The channels can be used as a concurrent property system
- ✅ Isolate the hardware code using channels
- ✅ Use channels as modules interface [in/out]

Zephyr® Project
Developer Summit

Zephyr® Project
Developer Summit

Questions & Answers

Thank you!