SCHEDULER VS EXECUTOR

Ruslan Arutyunyan, Mike Voss
Problem statement

People think in two protocols in terms of P0433:

• Executors
• Schedulers/senders/receivers

People often use the terms of schedulers and senders/receivers interchangeably.

From our perspective they are not the same things.
Example

From P0443 1.5.3 Scheduler:

- “Like executors, schedulers act as handles to an execution context.
- Unlike executors, schedulers submit execution lazily, but a single type may simultaneously model both concepts.
Example

From P0443 1.5.3 Scheduler:

• “Like executors, schedulers act as handles to an execution context.

• Unlike executors, schedulers \textbf{submit} execution lazily, but a single type may simultaneously model both concepts.
THE STATE OF THE ART
Execution context layer
Execution context layer

Execution Resources Interface

Resources abstraction layer

Execution Context

Execution Resources Interface
Execution context layer

- Execution Context

Resources abstraction layer

- Execution Resources Interface
- Scheduler

Work submission layer

- Work Submission Interface
- Work Submission Interface

Scheduler
CAN WE DO THE FOLLOWING?
Some method `scheduler()`?

**Execution context layer**

**Execution Resources interface**

**Resources abstraction layer**

**Work submission layer**

- Executor
- Sender
Apply the 5th item from P2235R0

From P2235R0:

• “5. Separate algorithms, most likely and preferably with names other than 'execute', can be provided in e.g. P1897 to allow straightforward fire-and-forget on schedulers or senders.”
std::execution::execute(ERI) for fire-and-forget

Execution context layer

Execution Resources interface

Resources abstraction layer

Work submission layer

Some method scheduler()? 

Sender