The Kernel

Scheduler: Round-Robin, Priority Queues, Tree Flavours
Scheduler Actors: Features, Timers, Async I/O
Streams Backends: Zero-copy, Message Passing
Linear Backends: Async I/O Disk Streams, Network Streams
Indexed Backends: Timers, Actors
Backpressured Message Bus/Buffers: Arc/Vec prealloc
Class: Low Latency, Real Time

Respect Kernels History

Richard Rashid. Mach 3. NUMA, Bus Oriented Components
Dave Cutler. Windows NT. True Async I/O on IoCompletionPort
BeOS. Travis Geiselbrecht. SMP Scaling of OS services
Microkernels: RT, Tiny codebases eCos/TRON, QNX, VxWorks
Unikernels: Erlang, Mirage, HaLVM

Stream/List duality

```rust
pub enum List<Message> {
        Nil,
        Cons(Message, std::marker::PhantomData<List<Message>>) }

pub struct Stream<Message> {
    head: Message,
       tail: Box<Stream<Message>> }

  pub trait Stream<Message> {
        fn head(&mut self) -> Message;
        fn tail(&mut self) -> Stream<Message>; }
```

Zero-copy and Message Passing

```
pub trait Future<Message,Error> {
        fn poll(&mut self) -> Result<Message,Error>;
        fn tail(&mut self) -> Future<Message,Error>; }

pub trait Process<Protocol, State, Error> {
        fn state(&mut self) -> State;
        fn send(&mut self, Protocol) -> Result<State, Error>; }

pub trait Discipline<Stream<Message>> {
        fn select(&mut self, u64) -> Stream<Message>; }
```
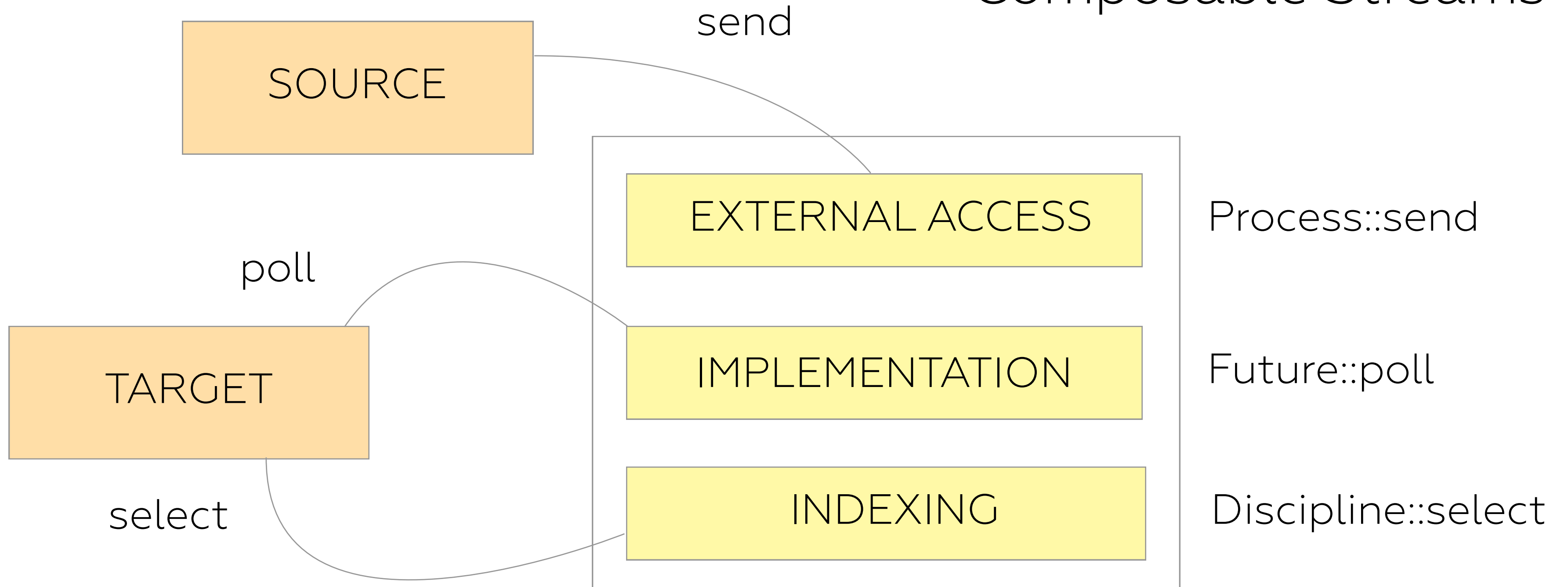
Order Processing Protocol

```
pub struct OrderState {
    state: OrderStatus,
    id: ID,
    price: Price,
    size: Size,
    side: Side, }
```

```
pub enum OrderProtocol {
    Request, Execute, Reject,
    UnsolicitedCancel,
    Cancel,
    Ack, Replace, }
```

```
type OrderProcess = Process<OrderProtocol, OrderState, Error>;
```
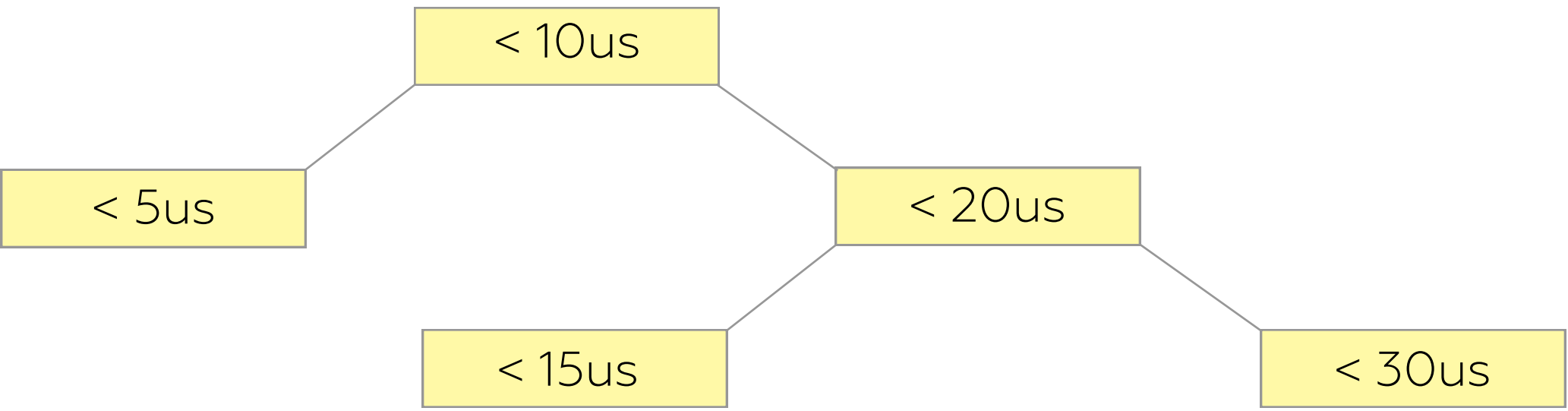
Composable Streams

SOURCE

send

EXTERNAL ACCESS

Process::send

TARGET

poll

IMPLEMENTATION

Future::poll

select

INDEXING

Discipline::select

Linear: MQ, EXT, DISK, NET
Trees: TIMERS
Priority Queues: TASKS, IRQ

Node Components

CPU # 1..3	SPU # 1

MQ	TIMERS	IRQ

TASKS	EXT

DISK	NET

schedulers

system streams
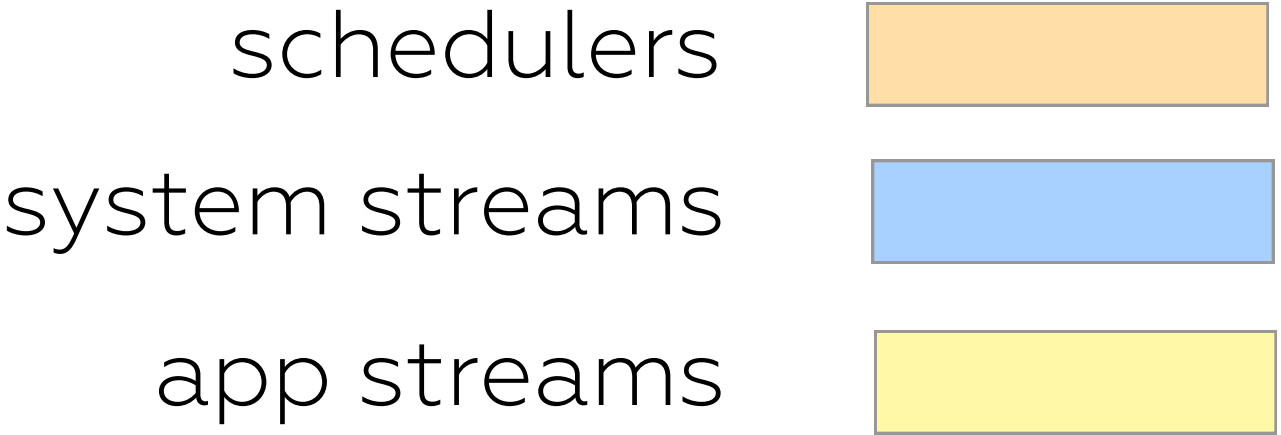
app streams
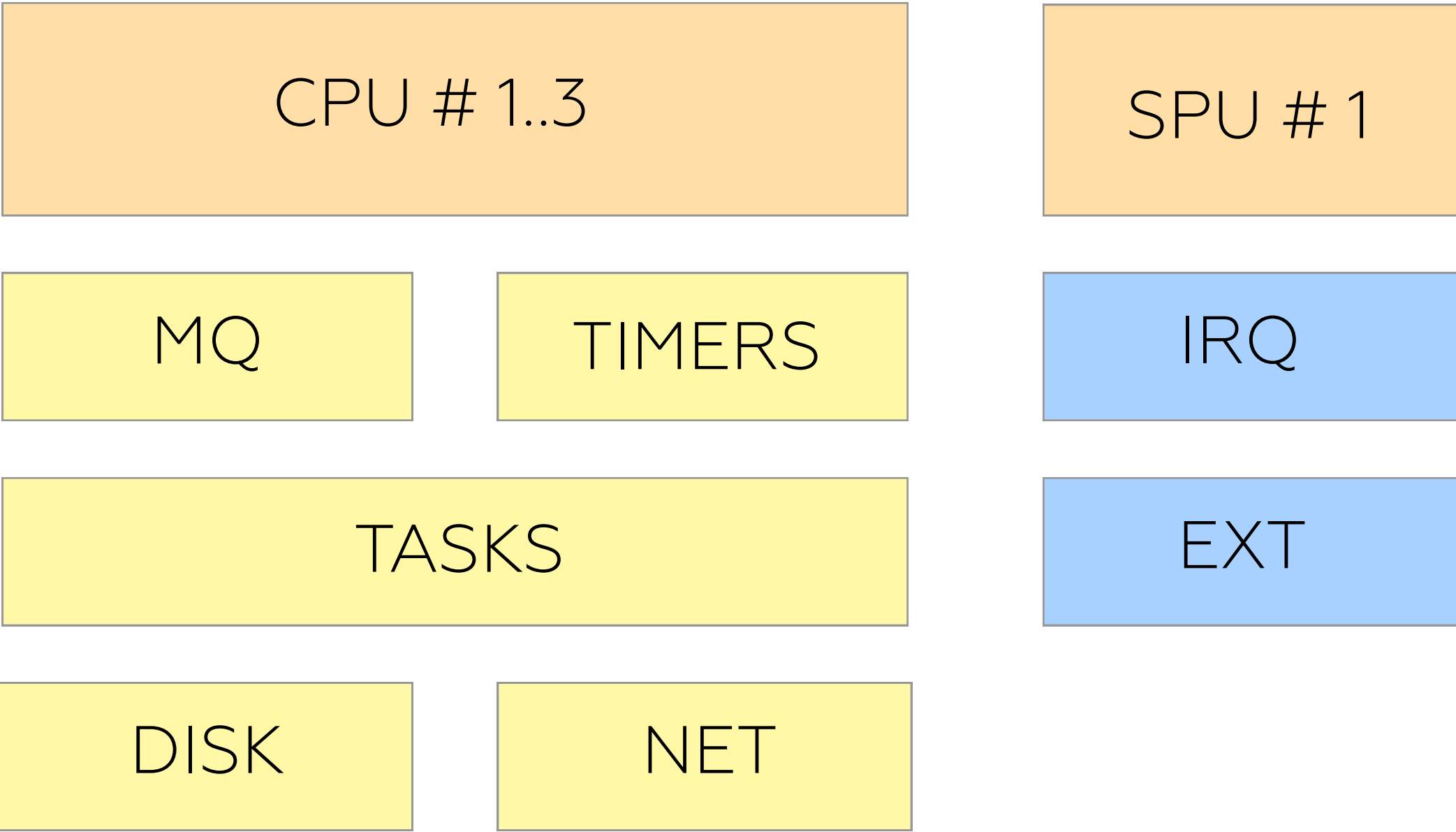
REACTOR

Discipline::select

TASKS

Core Context

TOKIO   REVENT

Single Task Tree per Core

NANOMSG

Scheduler::select

RX/RUST

Reactor Requested Features

1. Reactor Core Context
2. Task Context
3. Scheduleable Entities
4. Task List
5. Task Selection by Priority
6. API: spawn/select/poll
7. Inspirational Libraries: Rx, Nanomsg, Tokio

MQ

Vec, VecDeque, Link, Turbine, Pipes

Queue Types

CHANNEL

Discipline::select

SPSC/LINK

MPSC/SUB

SPMC/PUB

Non-schedulable Array Store

Queues Requested Features

1. Three Basic Drivers: SPSC, MPSC, SPSC
2. Different Backends: Vec, VecDeque, Link, Turbine, Pipes
3. Queue Properties
4. Queue with Priorities
5. API: create/pub/sub/link
6. Inspirational Libraries: Turbine, Rust Queues, Pipes

TIMERS

Discipline::select

TIMER

(interval , task_id)

Dedicated Core or Interrupt Queue

TOKIO

Scheduler::select

RX/RUST

Timers Requested Features

1. Timers Core Context
2. Timer Types: Oneshot Scalable Timers, Fixed List of Interval Timers
3. API: create/select/poll
4. Inspirational Libraries: Tokio, Rx

NET I/O

Vec

BUFFER

Socket Context

TOKIO

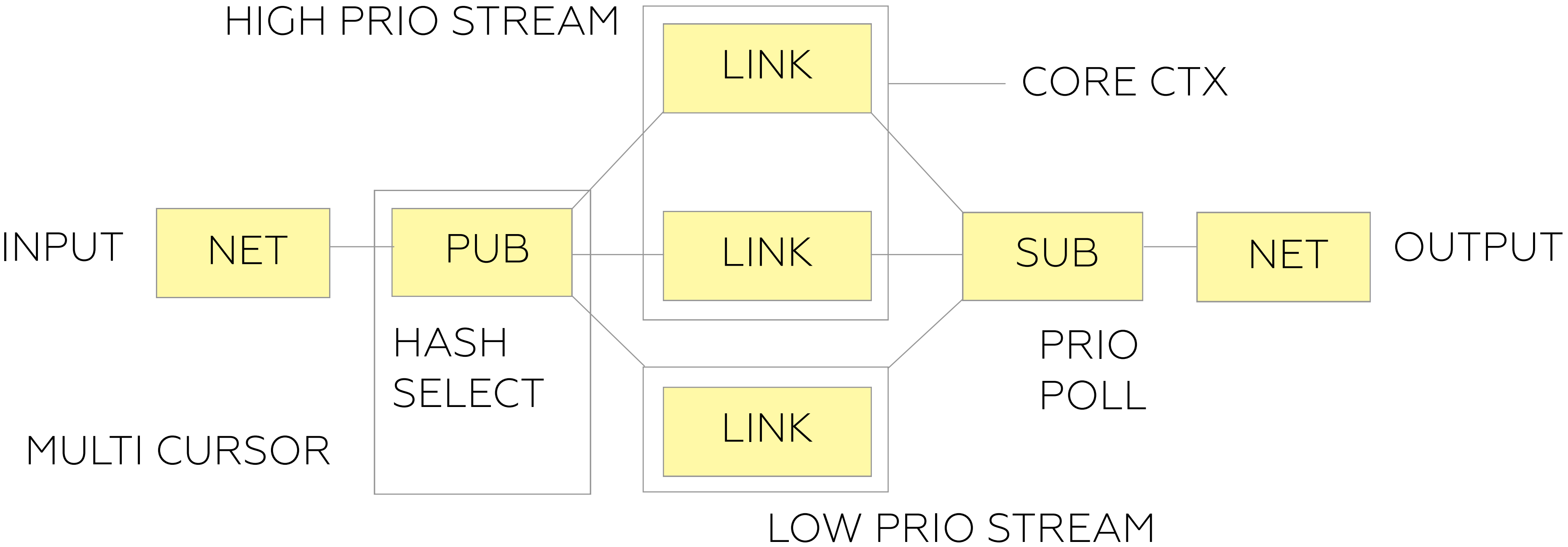Single Chain per
Core

NANOMSG

Scheduler::select

Network I/O Requested Features

1. Network I/O Core Context
2. Socket Context
3. Pipes/Queues Context and Interoperability
4. Timer Types: Oneshot Scalable Timers, Fixed List of Interval Timers
5. API: create/select/poll
6. Reconnections
7. Inspirational Libraries: Tokio, Rx

LOAD BALANCING CASE

Load Balancing
of Priority Streams per Core Buckets

HIGH PRIO STREAM

LINK — CORE CTX

INPUT — NET — PUB — LINK — SUB — NET — OUTPUT

HASH SELECT

PRIO POLL

MULTI CURSOR

LINK

LOW PRIO STREAM