Recording & Replay at HULKs

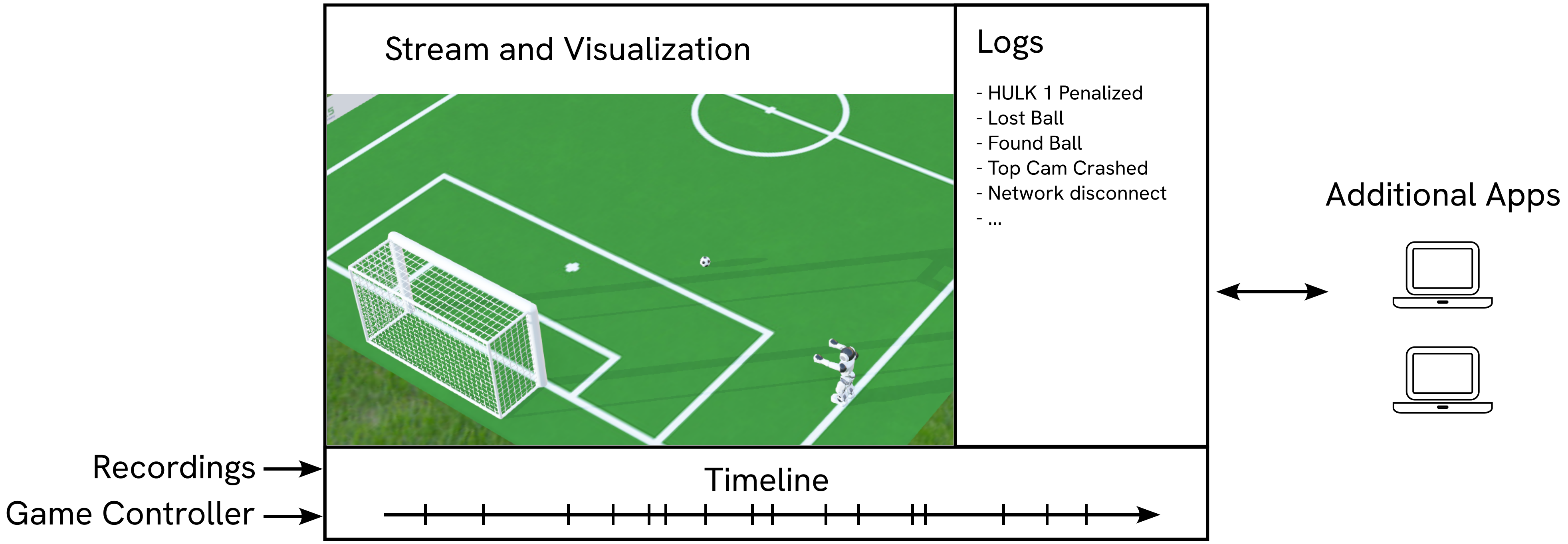
h3ndrk

RoHOW 2023

Motivation

- Debugging of rare situations is hard
- A lot of guessing in our post-game meetings
- No real insight
- @RoboCup 2023: localization recording & replay

Vision & Architecture



- HULK 1 Penalized
- Lost Ball
- Found Ball
- Top Cam Crashed
- Network disconnect
- ...

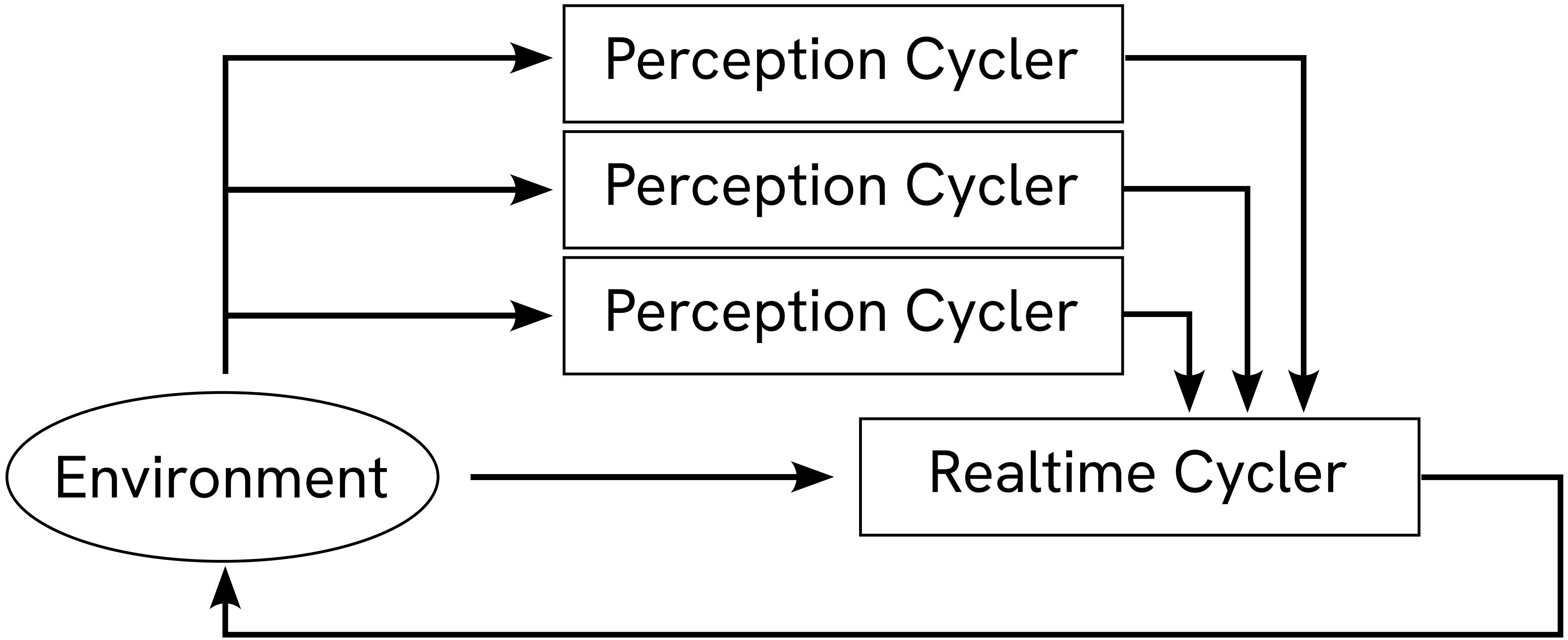
Vision & Architecture

- Game stream
- Game logs
- Replay
- GameController messages
- $\Rightarrow$ synchronized time

Trivial Approaches: Record all Outputs

- Recording all images?
  - 17-50 MiB per second
  - 1-3 GiB per minute
  - 10-30 GiB per game half
- On disk: not enough space
- On USB stick: too slow (~18 MiB per second)

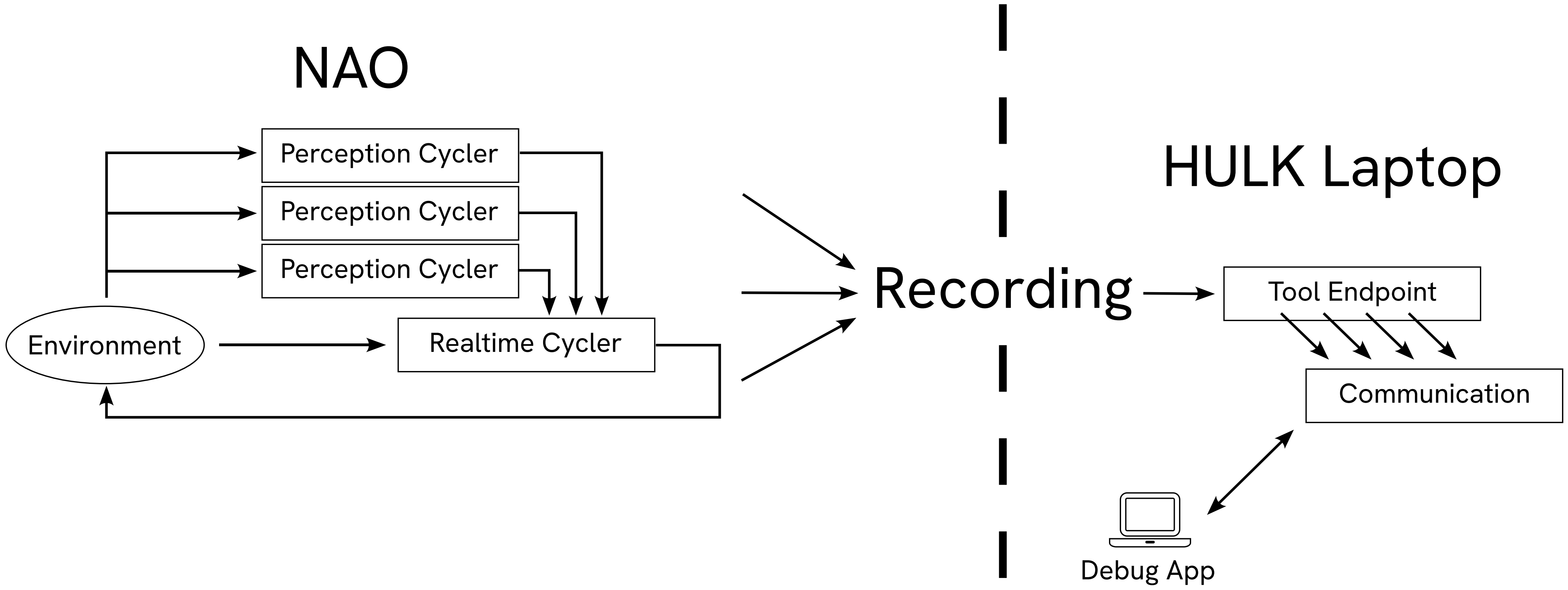
Recap: Framework & Robotics Domain



Implementation Idea

- Benefit of our framework and robotics code:
  $\text{framework} \cap \text{robotics} = \emptyset$
- Recording & replay $\rightarrow$ framework
- Record everything, analysis ideas might come later

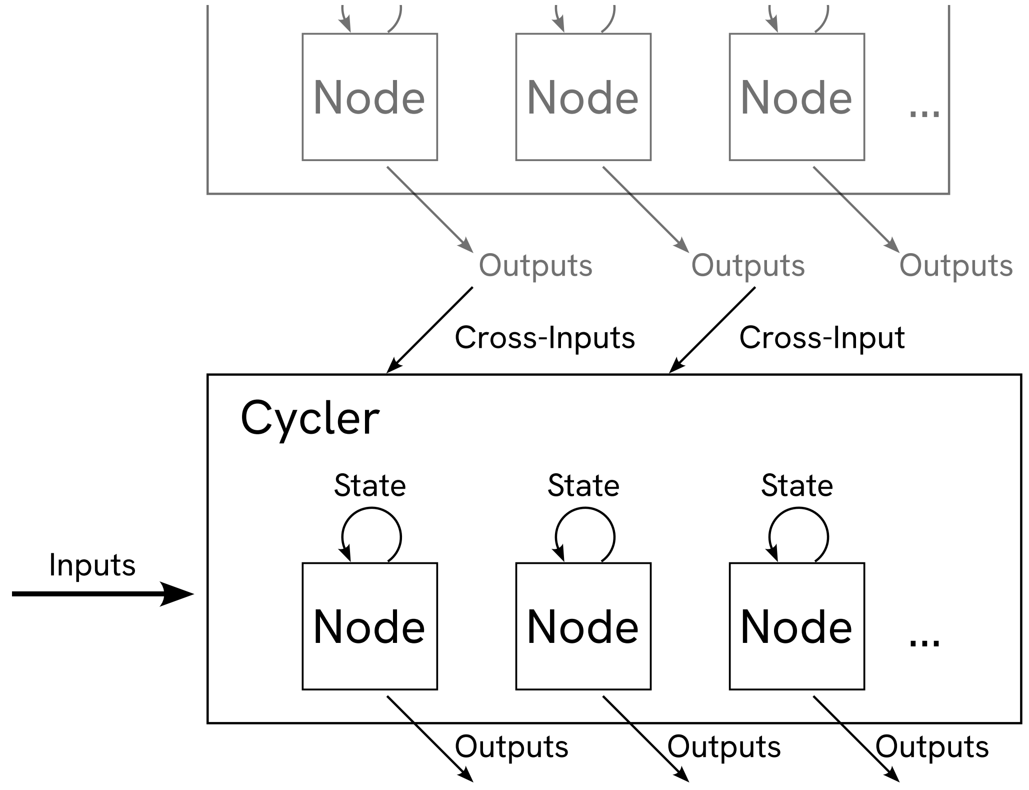
Architecture for Prototype



Recording

- Each cycler may record a frame
  - Triggers: communication, robotics, configuration
  - Variants: activation, burst, frequency
  - $\Rightarrow$ reduced frequency possible per cycler
- We want to replay single frames of cyclers
  - $\Rightarrow$ self-contained frames
  - Cycler inputs and node states are recorded
    - Assumption: Determinism in robotics domain

Implementation Idea



Replay

- Index allows random frame access
- Frames are self-contained, contain inputs & states
- Outputs are routed into communication
- Debug clients can see the replayed outputs

Implementation within the Cycle

Recording

- get own database
- **evaluate whether to record**
- **recording frame creation**
- run setup nodes
  - evaluate subscribed outputs, get parameters
  - run node
    - either evaluate main outputs by calling the nodes cycle or fill it with default
    - **record main outputs**
    - write main outputs into database
- announce future queue
- **record cross inputs**
- run nodes
  - evaluate subscribed outputs, get parameters
  - run node
    - **record node state**
    - either evaluate main outputs by calling the nodes cycle or fill it with default
    - write main outputs into database
- finalize future queue
- notify Communication
- **write recording frame**

Replay

- get own database
- iterate setup nodes
  - evaluate subscribed outputs
  - get parameters
  - **iterate node**
    - **write main outputs from frame**
- **read cross inputs into local variables (all following accessors use these)**
- run nodes
  - evaluate subscribed outputs
  - get parameters
  - iterate node
    - **restore node state**
    - either evaluate main outputs by calling the nodes cycle or fill it with default
    - write main outputs into database
- notify Communication

Current State

- Recording implemented
- Replay prototype nearly implemented
  - https://github.com/h3ndrk/hulk/tree/replayer

Challenges

- Reduction of data while still being complete
- Adaptions in code generation

Future

- Cache to disk and move to slower
- Foxglove Studio