Joining network-for-ML and ML-for-networks in a distributed orchestration approach

Holger Karl
Hasso Plattner Institute
University of Potsdam

A disclaimer instead of an agenda

- No results, just problems and ideas; invitation for discussions
- Goals and approaches of an incipient project
  - KIOps: Künstliche Intelligenz Operations
    - … specifically for Telco networks
  - BISDN, dfki, HPI, Infosim, KIT, T-Labs

- Two aspects for today
  - Is there a difference between networks-for-AI/ML and AI/ML-for-networks?
    - For training, inference?
  - How to scale up, be resilient?

To clarify: Networks for ML ⇔ ML for networks?

- Networks for ML
  - Set up networks such that ML training and inference are well supported
  - Can pertain to routing, traffic engineering, congestion management, placement of functions, …
- ML for networks
  - Use ML approaches to operate networks
  - E.g., to improve traffic engineering, congestion control, orchestration problems, …
    - Across many time scales, from fast control loops to combinatorial optimization problems

Common claim: Very different things!

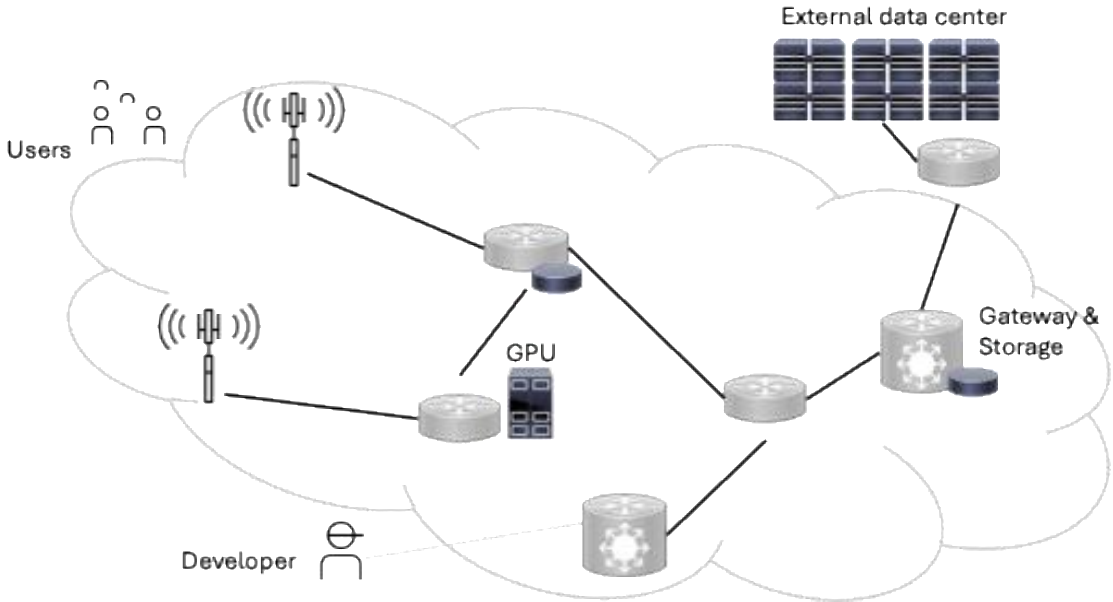
- But are they?

- An ML-based application collects data, trains models, infers actions
  - Immaterial, in principle, where data comes from, what these actions are
  - No matter whether inferred action is an autocompletion or a routing table change
  - … with obvious caveats

- Challenge: Find a structure where both types of ML applications (ML4networks, networks4ML) can be operated in the same system, without getting in each other ways

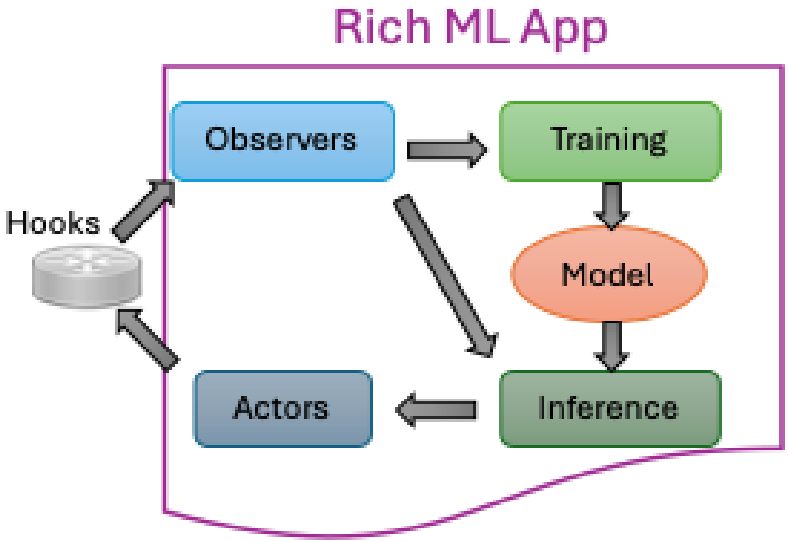
An orchestration problem: Ressources

- Here: ML applications in telco networks
- Hence: Many different types of ressources



An orchestration problem: Applications

- An ML application should have access to which data sources, which control actions, which model stores?
- Applications *hook into* their environment, with explicit control over these hooks

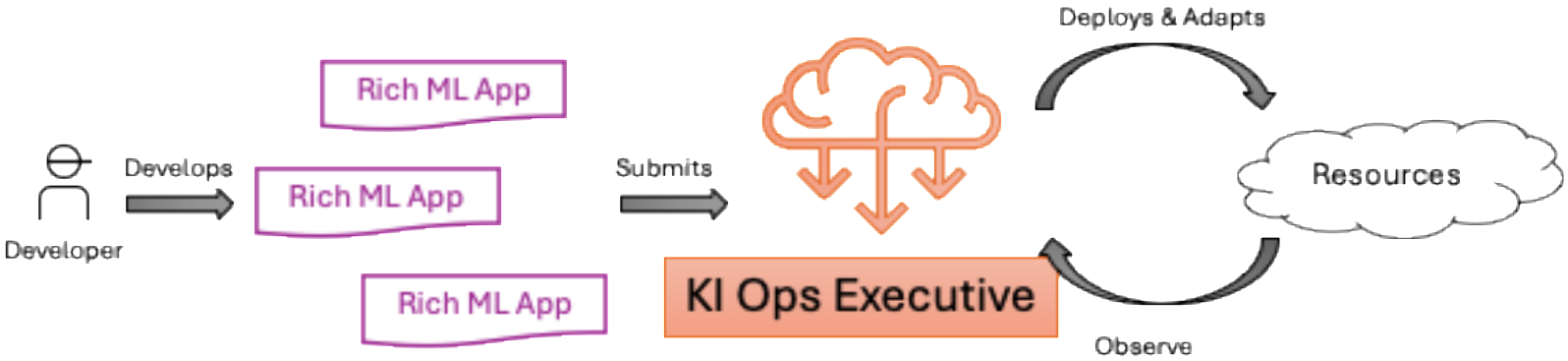


An orchestration problem: Annotated applications

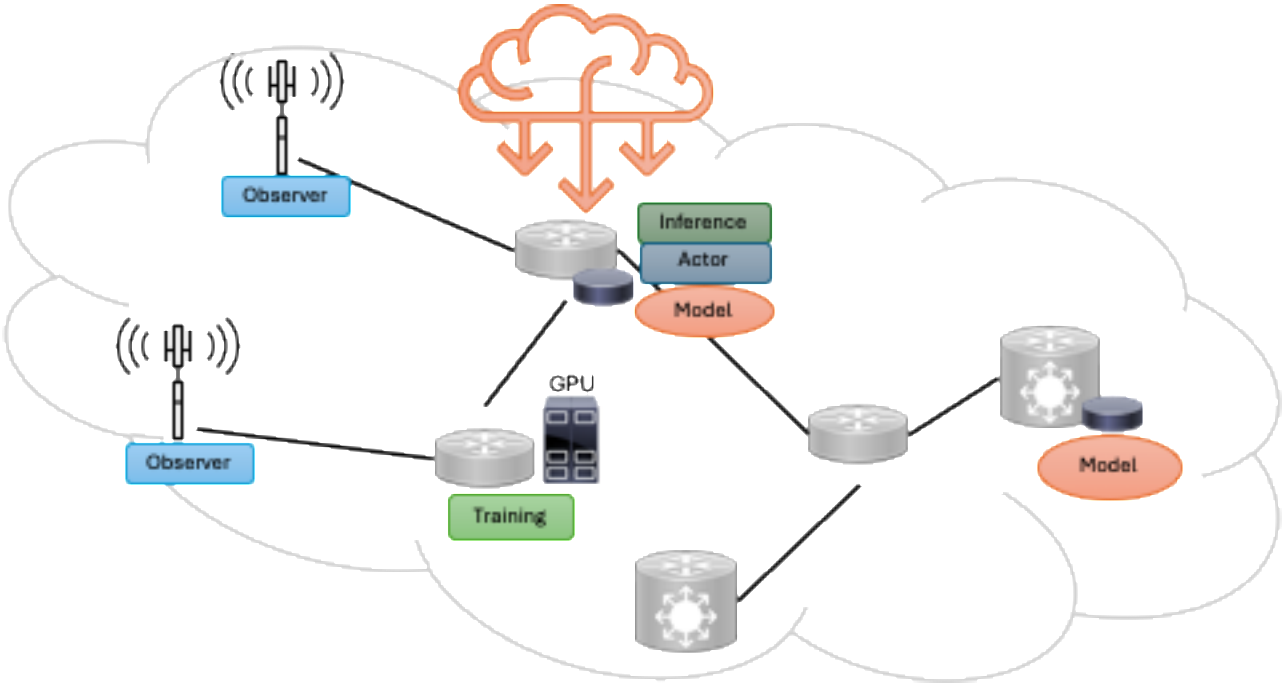
- Hard to know myopically; needs explicit information
  - Inspiration: Meta data for applications in other fields, eg. Microservices or NFV
  - Nick name: “Bill of material”
- Could be provided by application developer
- Should not be trusted, but verified by operator

```
app: beamforming
  training:
    observer: read-channel-statistics.sh
      obseration_spots: basestation[*]
    retraining_period: 1d
    training_pattern: parameter_server
    training_algorithm: PPO
      shared_obserations: true
    model: beamforming-model  # Training output
  inference:
    model: beamforming-model  # Input for inference
      reusable: true  # Multiple copies can be used
    observer: read-current-channel-status
      period: 20 ms
      size: 512 bytes
      observation_sites: basestation[*]
    actions: set-beamforming-parameters.sh
      acts_on: basestation[*].precoding_matrix
      # enables conflict detection
      deadline: 5ms # from observation to action
      size: 1024 bytes
    inference:
      compute: 100 FLops
      memory: 20 KBytes
  AAA:
    certificate: xie6eeP7
    privileged: true
```

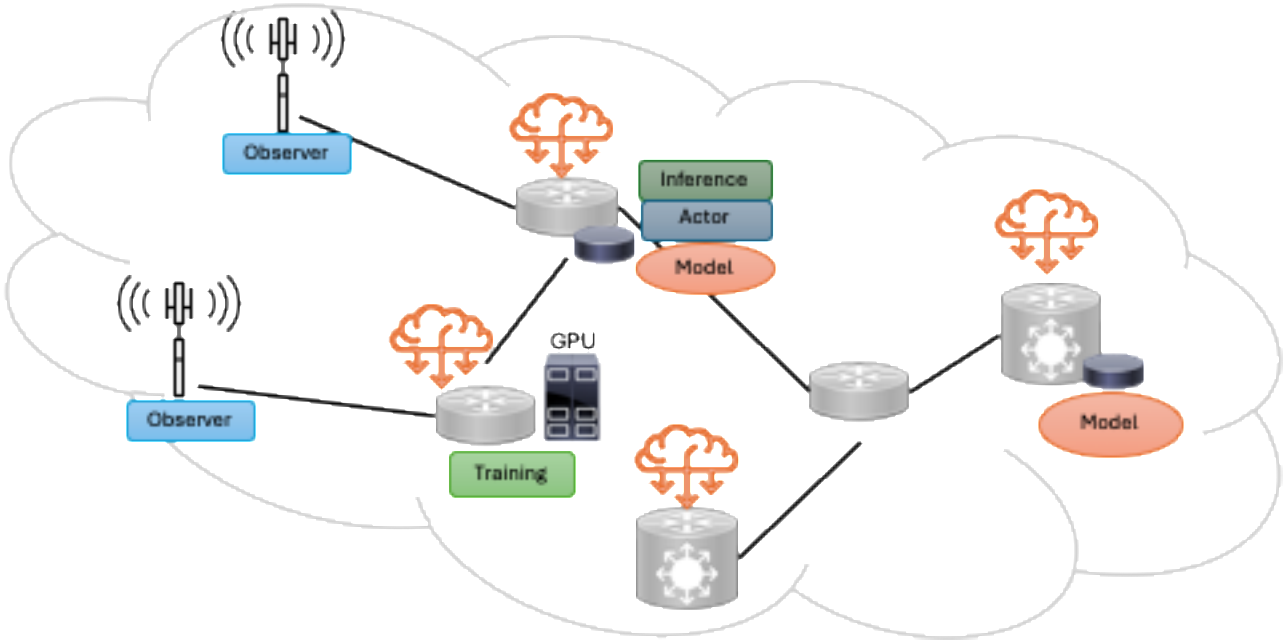
Possible workflow



A single brain?



Scaling out the brain?



Obvious multi-brain challenges

- For us: Whether and how to keep multiple brains in sync
- Organization?
  - Hierarchies? Overlapping or strict?
  - Or peer-to-peer-type communication patterns?
  - …?
- Consistency?
  - Which data structures need to be kept how consistent?
  - Where should we be on the CAP triangle?

KIOps: More aspects

- How to discover resources (hooks for observation or actions), figure out access rights?
- What are useful, yet extensible formats for Bills of Material?
- What are promising business models?
- …?

Thanks!

- Please do feel free to reach out
- Easiest: holger.karl@hpi.de