DDFlow*: Visualized Declarative Programming for Heterogeneous IoT Networks

Joseph Noor, Hsiao-Yun Tseng, Luis Garcia, Mari Srivastava

***Not the DDFlow that appears in Google searches**

So Many (Io)*Things*

- The IoT world has a large variety of devices with wide-ranging capabilities

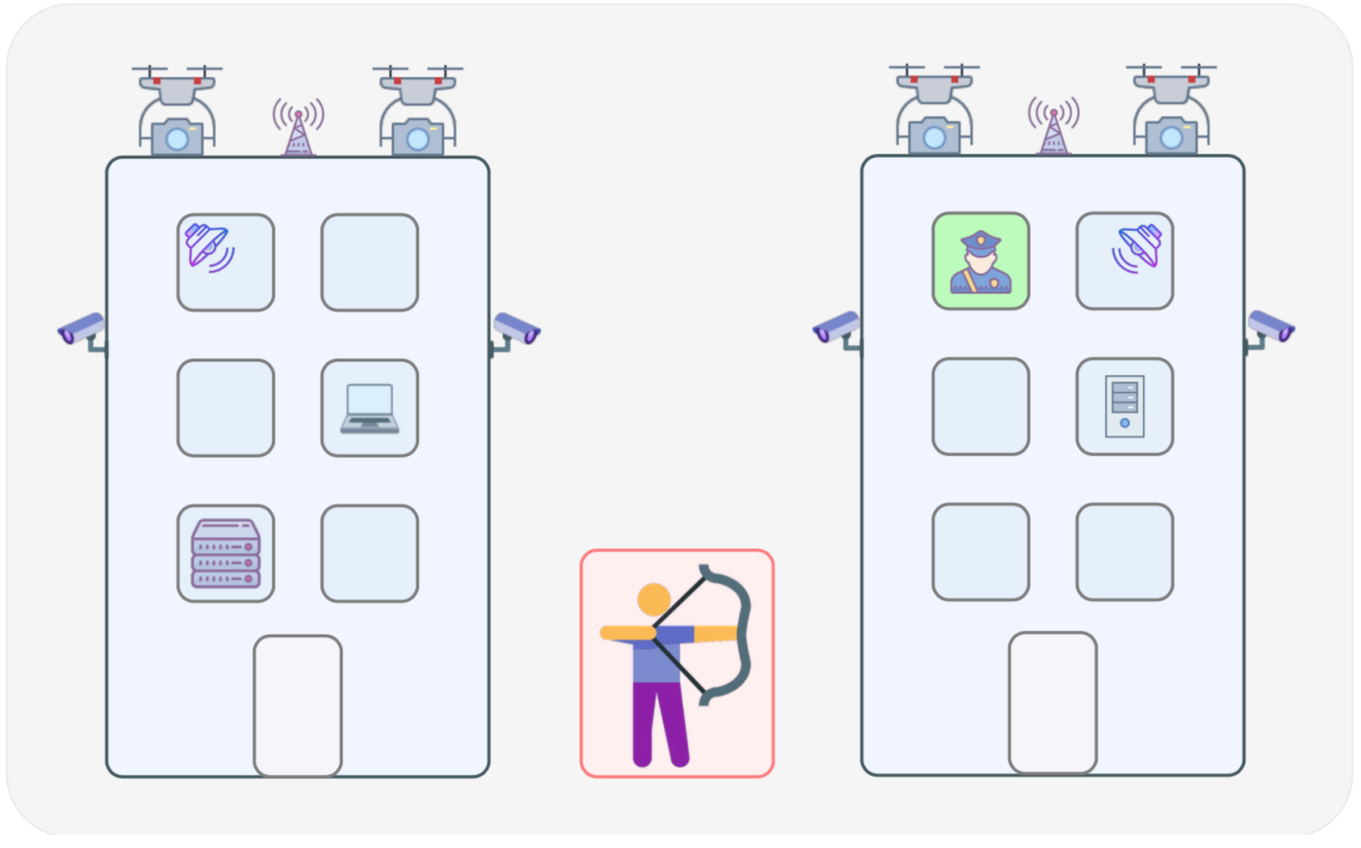
- **Challenge**: effectively specifying and managing coordinated activity across these heterogeneous hardware

    - **Added challenge**: don’t worry about low-level network, hardware, and coordination details

- Tools like DFuse, Kairos, Regiment, Mobile Fog, and D-NR **lack** either good declarative specification language, fault tolerance, dynamic adaption desired

Motivating Example

- Recruit cameras in region to identify objects-of-interest
- Classify captured image frames to detect target
- Speaker plays sound upon detection
- Drones pursue target and stream live feed

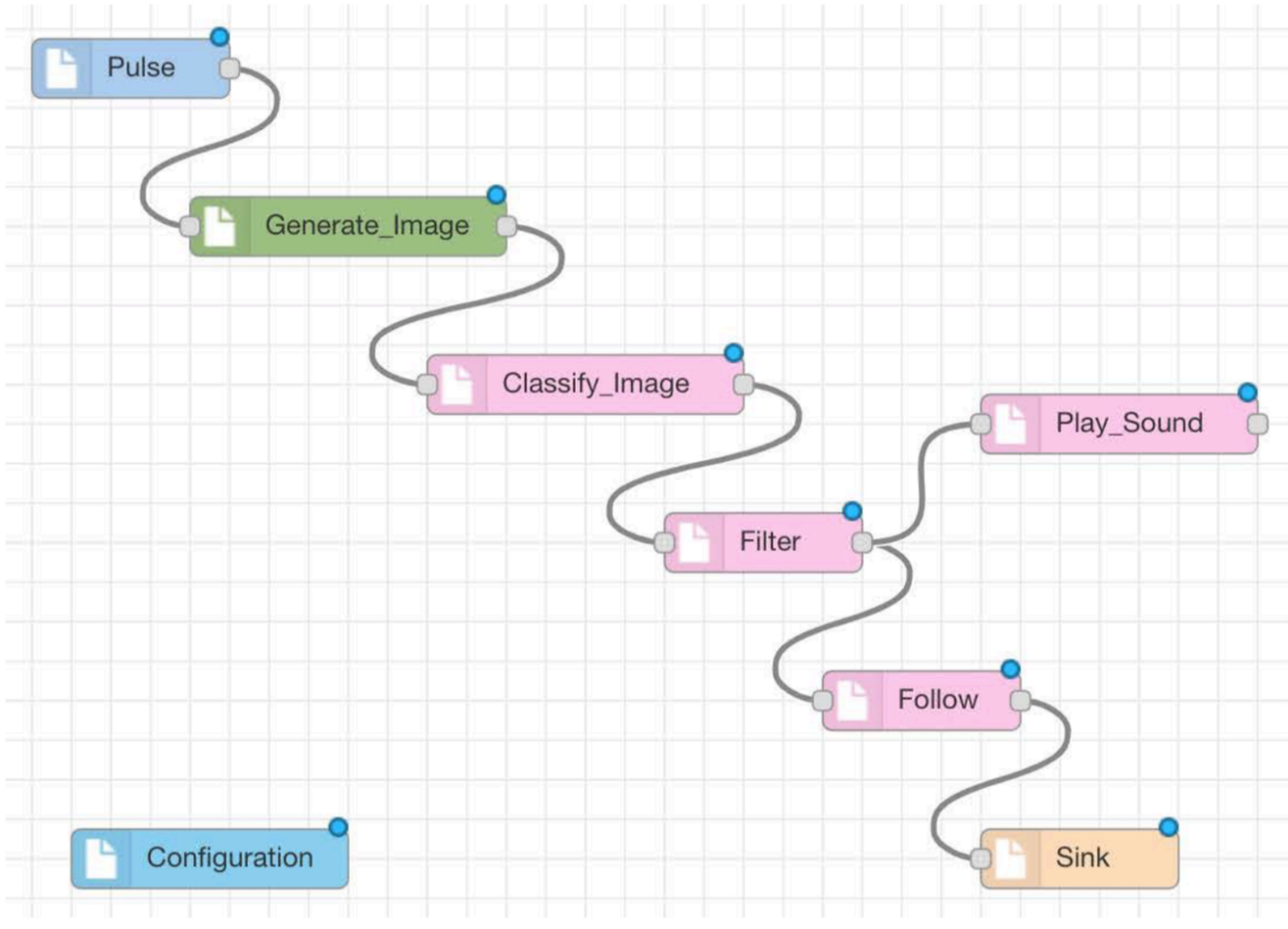


DDFlow

- Visual declarative macroprogramming* abstraction, on top of Node-RED

- Runtime for scaling and adapting deployments to diverse environments

- System converts specification into computation graph and deploys across network

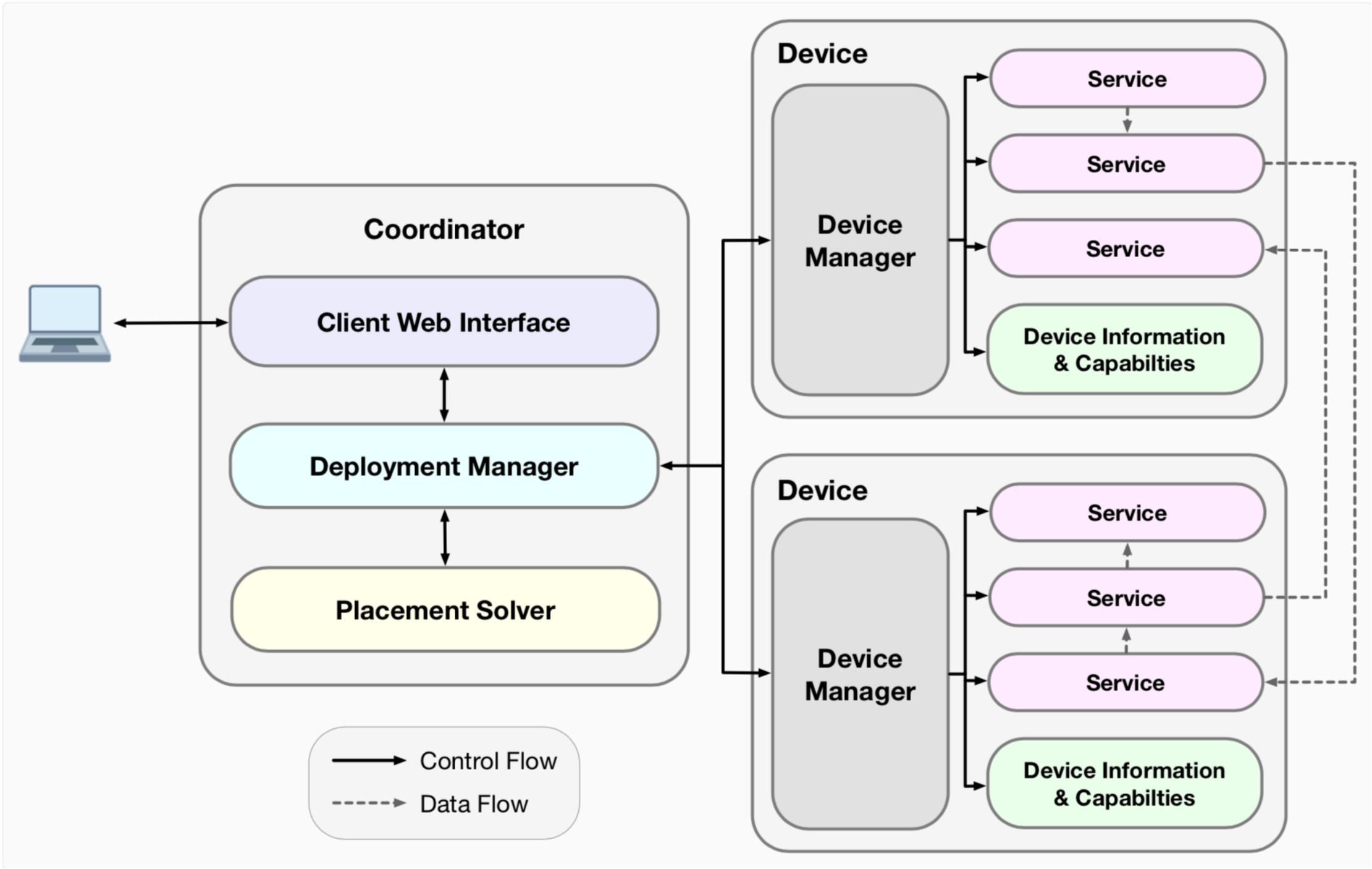


***What a bad name**

DDFlow Primitives

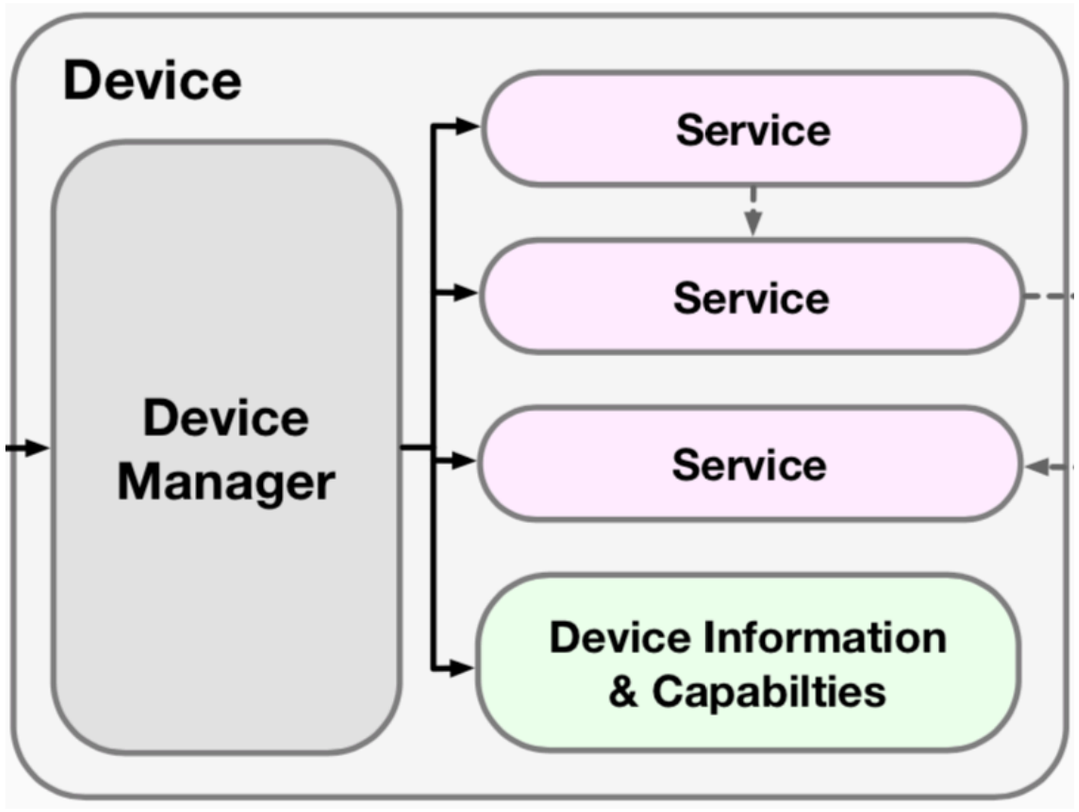
- **Node**
  - A stateful function that maps inputs to outputs
  - Corresponds to task instantiation that must be deployed onto a device
  - Optionally parameterized by *Region* (some type of location information) and *Device* (precise placement)
- **Wire**
  - A connection in the dataflow graph
  - Carries a key-value dictionary from output Node to input Node
  - *Stream* (one-to-one), *Broadcast* (one-to-many), *Unite* (many-to-one)
- Extensions
  - Can create classes implementing the DDFlow interface to make **new nodes**

System Architecture



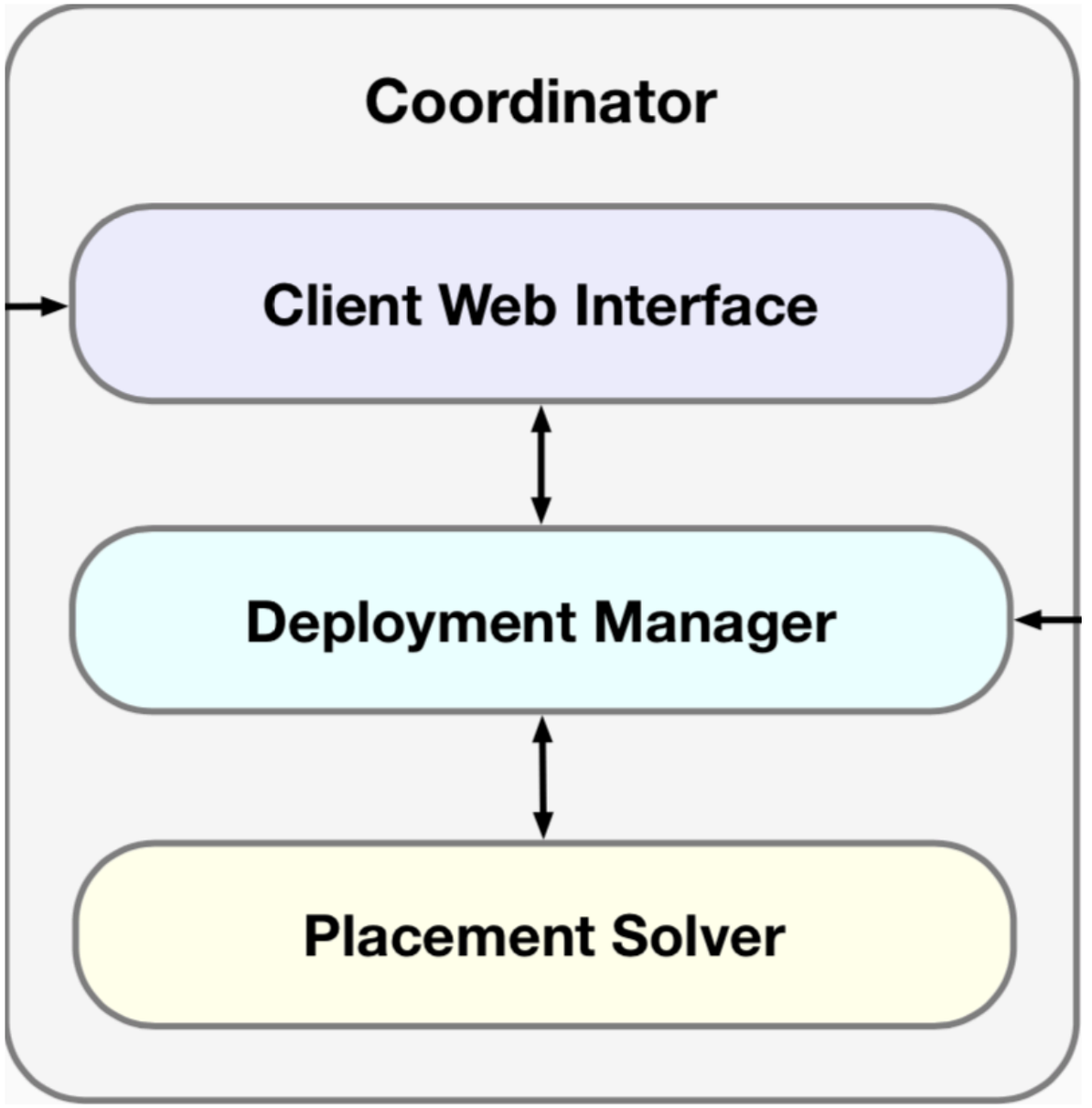
System Architecture

- Service-Oriented Architecture (SOA)
  - Cloud to edge all present same *Service* high-level interface
  - Services like image classification, playing sound
- Intra-device coordination via **Device Manager**
  - Web server runs on every device (or proxy)
  - Exposes device details
    - Location
    - Utilization
    - Estimated service/network latencies
    - Devices within wireless range



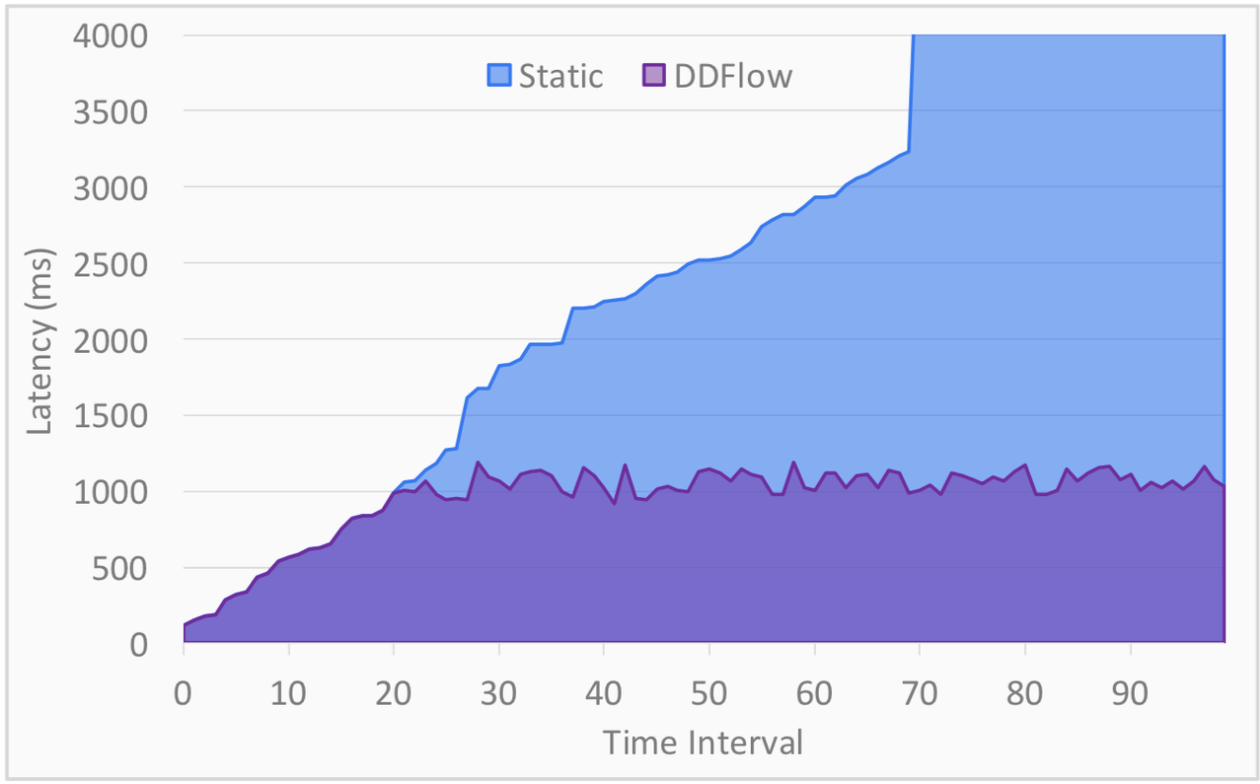
System Architecture

- **Coordinator** is a web server accepting/managing applications as issued
- Constructs network topology graph and task graph
- Detects significant network and compute changes and adjusts deployment (remapping)
- **Placement solver** maps an application task graph to available devices
  - Linear programming to minimize longest path's end-to-end latency
  - Constraints:
    - Task graph neighbors are accessible in network
    - Devices have implementations and resources ecessary for task



Evaluation

- Simulated motivating application with Airsim and Minimet
- Device Overload
  - Static deployment streams to fastest classifier but can't handle overload; eventually crashes*
  - DDFlow switches to different classifier upon latency increase
- Access Point Failures
  - Static deployment can't establish routing path from drone to client on AP failure
  - DDFlow switches to a WiFi ad-hoc peer-to-peer protocol



Adaptation during device overload