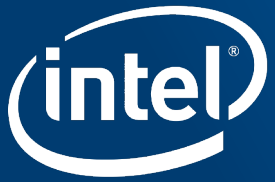


EDGE APPLICATIONS: DEVELOPMENT OF STANDARDS SUPPORTING AN OPEN EDGE COMPUTING ECOSYSTEM

Michael McCool

Principal Engineer, Intel
W3C Web of Things WG Co-chair

OUTLINE

- **Motivation**
  - Top-Down vs. Bottom-Up Deployment Models
  - Use Cases
  - Definition and Goals
- **Possible Technical Approaches**
  - Discovery
  - Compute Service Offload
  - Orchestration Service Installation and Management

OUTLINE

- **Motivation**
  - Top-Down vs. Bottom-Up Deployment Models
  - Use Cases
  - Definition and Goals
- **Possible Technical Approaches**
  - Discovery
  - Compute Service Offload
  - Orchestration Service Installation and Management

EDGE COMPUTING: TOP-DOWN VS. BOTTOM-UP DEPLOYMENT

1. Extend Cloud Computing Down

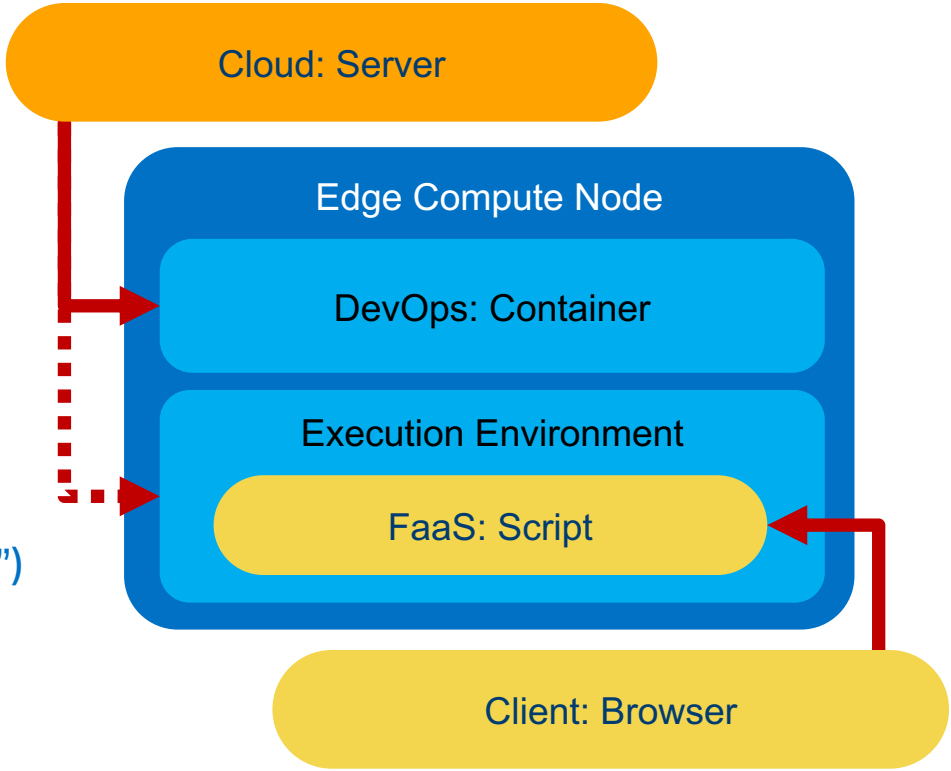
- Services managed ***by provider***
- **General** execution environment
- "DevOps": based on containers

2. Extend Web Computing Up

- Apps accessed ***by user*** via links
- **Specialized** execution environment
- "FaaS": based on functions (typ. "scripts")

Complementary

- 1 is the foundation for 2



USE CASES FOR USER-CENTRIC (BOTTOM-UP) DEPLOYMENT

Smart Retail

- Small business owners self-managing technology (1)
- Large retail franchises deploying applications for use on employees' own devices (BYOD context)

Smart City

- 40% of smart city use cases require multivendor solutions (2,3)
- Cities need to develop third-party app ecosystem to best provide value to citizens

(1) https://www.conexxus.org/
(2) https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world
(3) https://machinaresearch.com/news/smart-cities-could-waste-usd341-billion-by-2025-on-non-standardized-iot-deployments/

GOALS

Primary:

- Define “compute utilities” supporting **client-managed** “edge worker” services
- Allow clients to **offload compute** to “nearby” compute utility infrastructure
- Allow clients to **manage distributed IoT orchestration**

Secondary:

- Support secure, monetized, differentiated (e.g. accelerated) edge computing services
- Support development of a third-party application ecosystem
- Extend web programming to simplify development and deployment of applications
- Allow clients (users) to easily and dynamically find compute utilities (discovery)

TARGET CAPABILITIES AND THEIR REQUIREMENTS

Capability 1: Compute Offload

- Allow browser-based applications, small IoT devices, and client computers access to accelerated compute utility
- Compute utility may be on-board (device), local (edge), or remote (cloud)

➢ **Requirement:** Access to accelerated computing (GPU, FPGA, NN-ASIC, etc)

Capability 2: IoT Orchestration

- Install programmed orchestration function for derived IoT services

➢ **Requirement:** Access to local network and IoT devices

➢ **Requirement:** Persistent installation and event-driven execution

Other General Requirements

➢ **Privacy:** Trusted information and metadata management

➢ **Security:** Integrity, confidentiality, access control, authentication

➢ **Discovery:** Local and remote, devices and services, open but protected

➢ **Management:** Installation, cancellation, monitoring, payment

EDGE COMPUTING AND IOT ORCHESTRATION

Better Together!

- **Compute Offload** (by itself) lacks access to sensors and actuators
  … interesting applications use data to make decisions and take actions
- **IoT Orchestration** (by itself) lacks capability to make complex decisions
  … complex decisions need compute-intensive analytics

IoT orchestration + Edge Computing have *many* applications

- **Security:** motion sensor, camera, *person detection*
- **Inventory:** door open sensor, *product identification*
- **Logistics:** location tracking, 3D scanning, camera, *path planning*
- **Energy:** temp sensor, heater control, *person detection*, *machine learning*
- **Marketing:** door sensor, proximity sensor, camera, *sentiment analysis*
- **Cleaning:** robot vacuum cleaner, *obstacle classification, path planning*

OUTLINE

- Motivation
  - Top-Down vs. Bottom-Up Deployment Models
  - Use Cases
  - Definition and Goals
- **Possible Technical Approaches**
  - Discovery
  - Compute Service Offload
  - Orchestration Service Installation and Management

SUMMARY OF PROPOSED TECHNICAL STANDARDS STRATEGY

Extend PWAs, Service Workers, and Web Workers

- Web Workers extended to “Edge Workers”, supporting remote install on compute utilities, persistent lifetimes, event-driven execution, accelerated computing (e.g. via WebNN, TensorFlow.js, etc.), and to the local network for IoT orchestration
- PWAs/Service Workers extended to “Edge Apps”, supporting management lifecycle and remote “Edge Worker” components on compute utilities
- Use of WASM to package Edge App components offloaded to compute utilities.

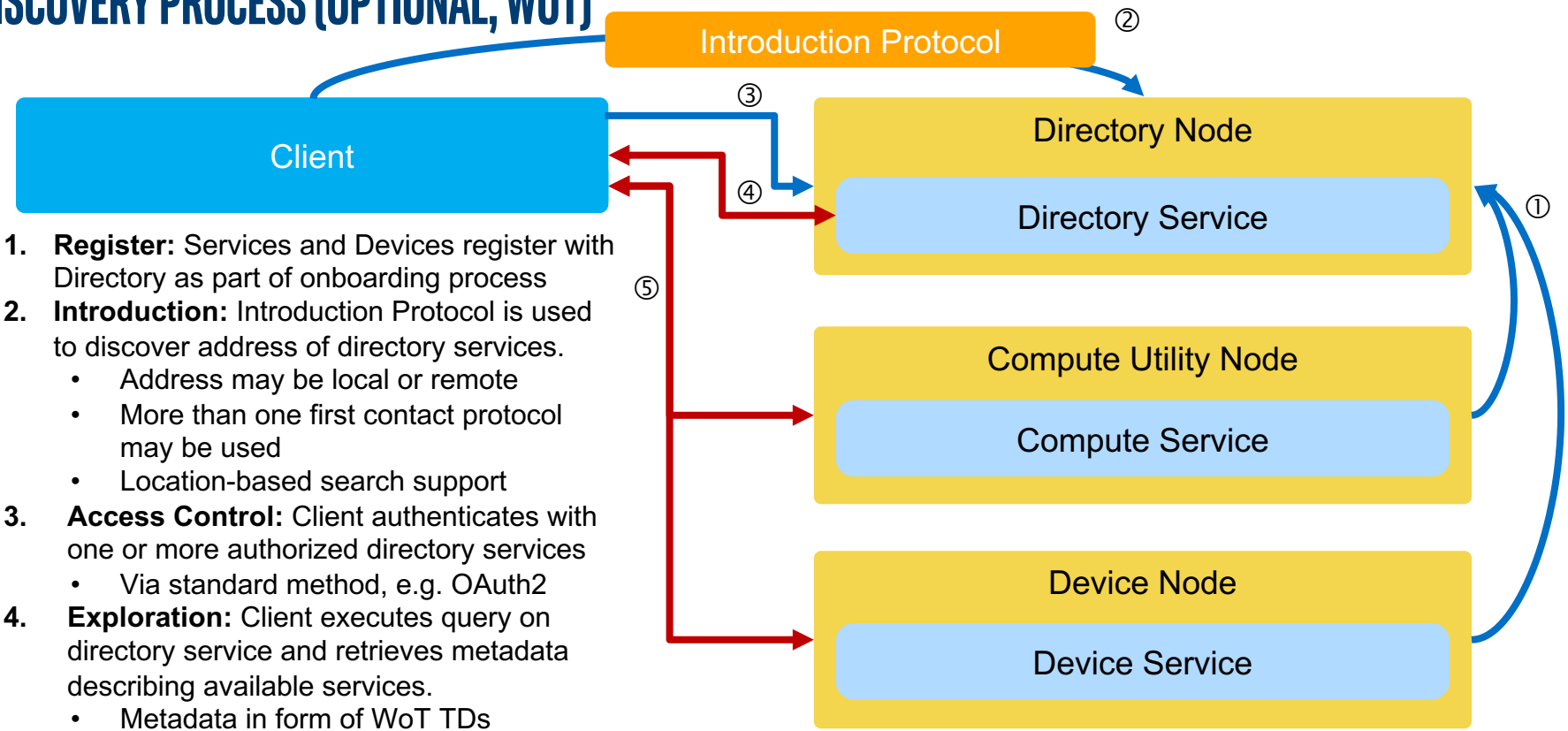
Extend Web of Things

- Extend WoT Discovery (WIP) to also apply to compute utilities
- Support IoT orchestration via WoT Scripting API (WIP) in Edge Workers

New Standards Development

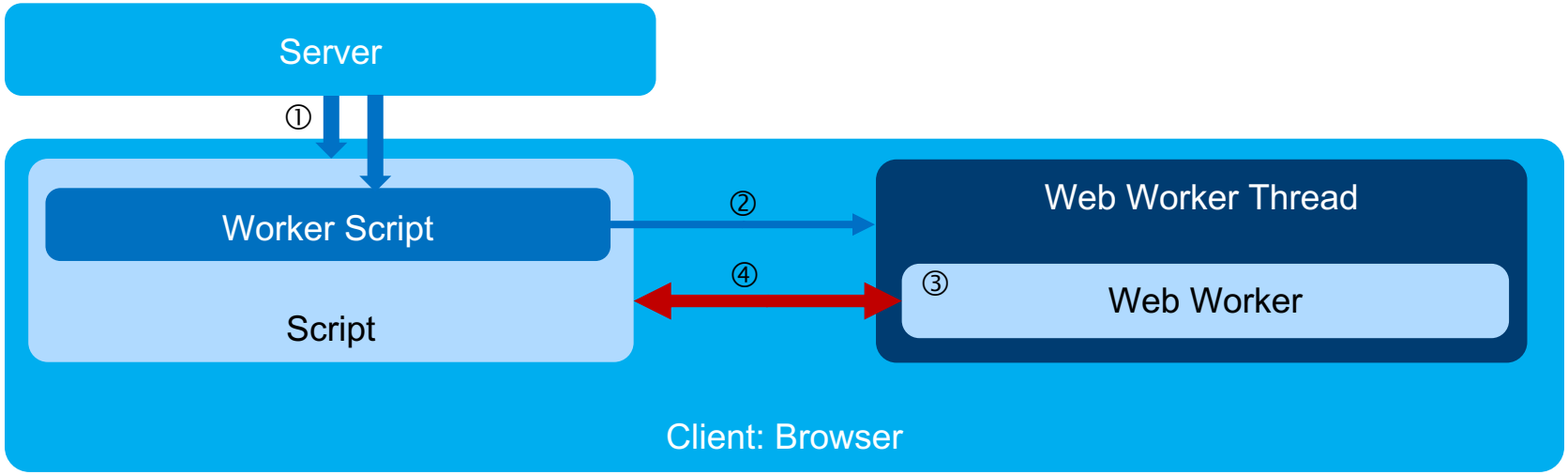
- Standardized Management API for compute utilities

DISCOVERY PROCESS (OPTIONAL; WOT)



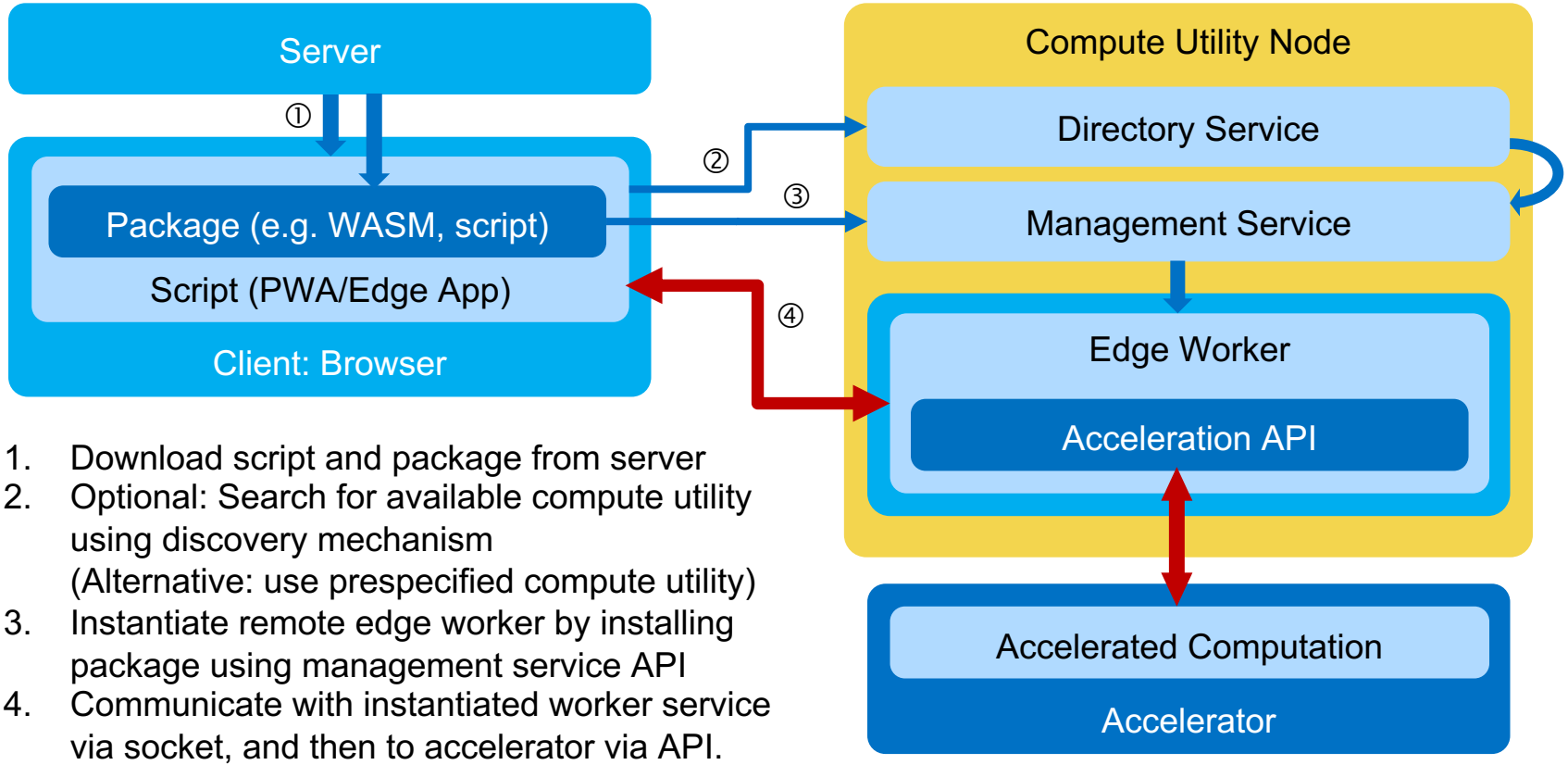
1. **Register:** Services and Devices register with Directory as part of onboarding process
2. **Introduction:** Introduction Protocol is used to discover address of directory services.
   - Address may be local or remote
   - More than one first contact protocol may be used
   - Location-based search support
3. **Access Control:** Client authenticates with one or more authorized directory services
   - Via standard method, e.g. OAuth2
4. **Exploration:** Client executes query on directory service and retrieves metadata describing available services.
   - Metadata in form of WoT TDs
   - Semantic search support possible
5. **Use:** Client accesses services.

EXISTING: COMPUTE OFFLOAD VIA WEB WORKERS



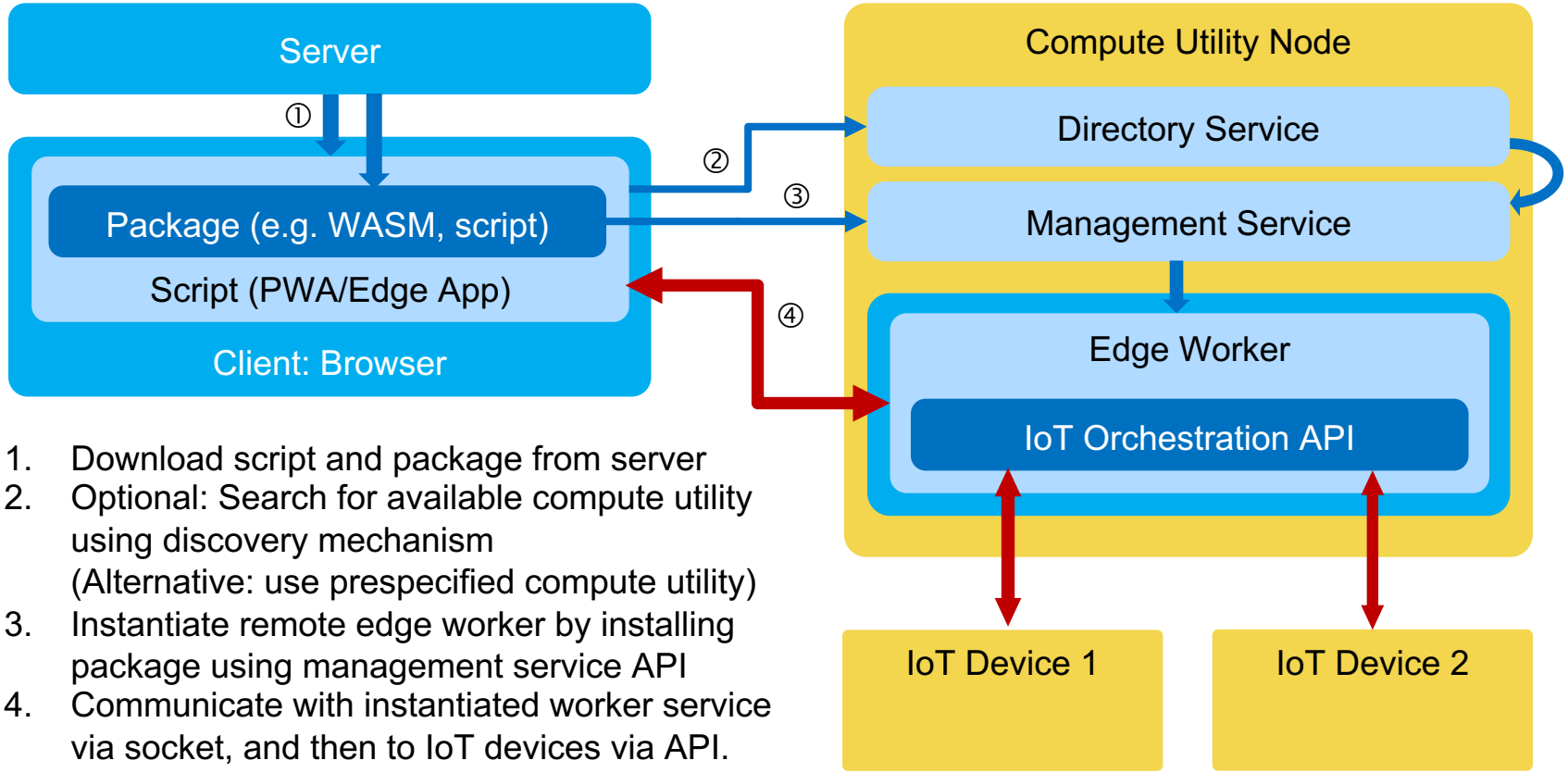
1. Fetch script and worker package from server
2. Create web worker and offload worker script.
3. Worker instantiated on separate thread inside client, runs in parallel with main thread
4. Main thread communicates with instantiated worker service to retrieve results

PROPOSED: COMPUTE OFFLOAD VIA EDGE WORKERS



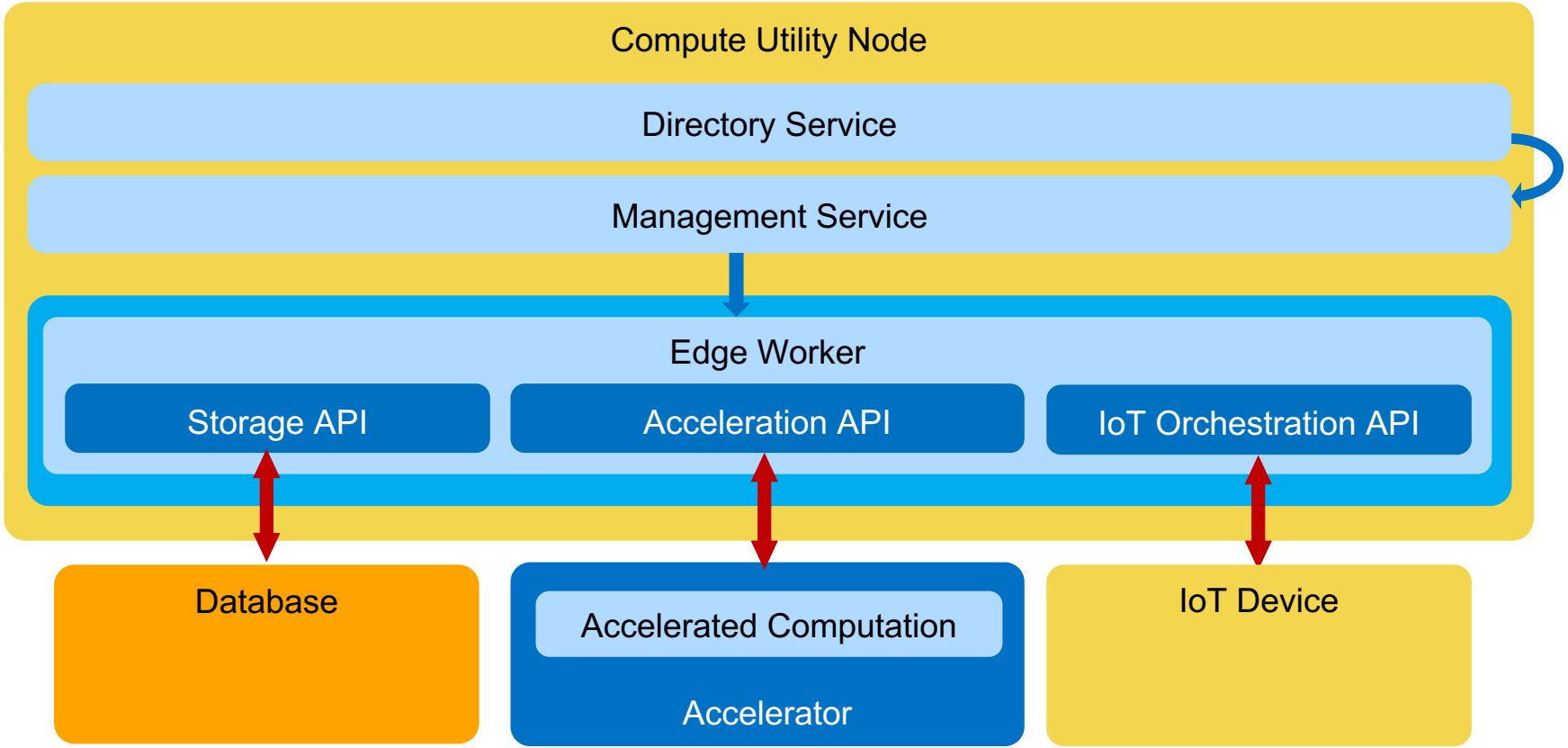
1. Download script and package from server
2. Optional: Search for available compute utility using discovery mechanism
   (Alternative: use prespecified compute utility)
3. Instantiate remote edge worker by installing package using management service API
4. Communicate with instantiated worker service via socket, and then to accelerator via API.

PROPOSED: IOT ORCHESTRATION VIA EDGE WORKERS (AND PWAS/EDGE APPS)



1. Download script and package from server
2. Optional: Search for available compute utility using discovery mechanism
   (Alternative: use prespecified compute utility)
3. Instantiate remote edge worker by installing package using management service API
4. Communicate with instantiated worker service via socket, and then to IoT devices via API.

MULTIPLE APIS ACCESSED FROM EDGE WORKERS



REQUIRED STANDARDS DEVELOPMENT

Management API (network and scripting) to Instantiate Workers

- API for a compute service that allows installation of a packaged worker

Packaging and Worker Management

- Worker encapsulation that allows installation in a sandboxed and isolated environment with all their dependencies and suitable (but controlled) access to other services. Options: WASM, scripts, containers.

APIs for Compute Acceleration (e.g. WebNN) and IoT Device Access (e.g. WoT)

- Orchestration services need to access other IoT devices
- Compute services need access to accelerated compute capabilities
- Installation of edge workers should be possible from browser and web application contexts, e.g. as extension of PWAs and/or web workers

Optional: Discovery (network and scripting API)

- Find a compute utility that can host a worker (requirements-based search)
- Can be an extension/application of WoT Discovery process