Load Balancing for Interdependent IoT Microservices

Ruozhou Yu, Vishnu Teja Kilari, **Guoliang Xue**

Arizona State University

Dejun Yang

Colorado School of Mines

Outlines

Background and Motivation

System Modeling

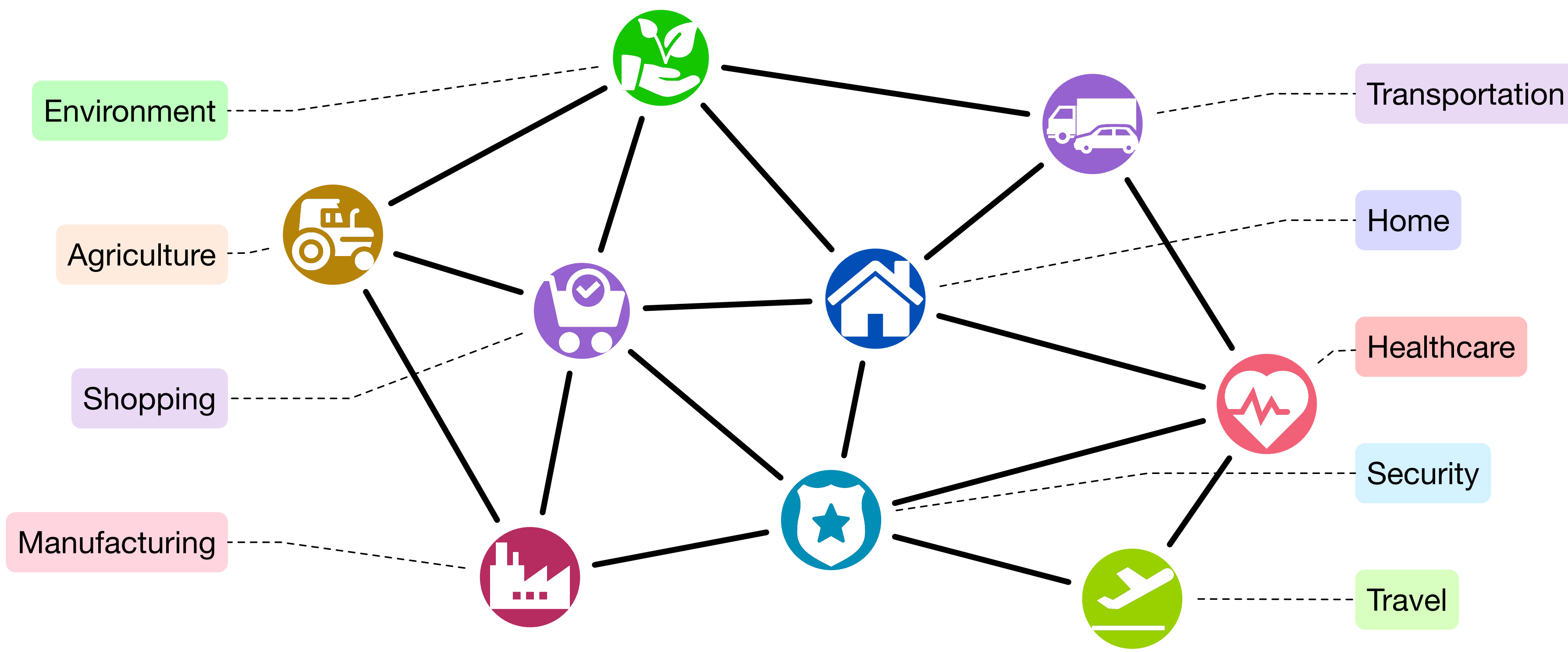
Algorithm Design and Analysis

Performance Evaluation

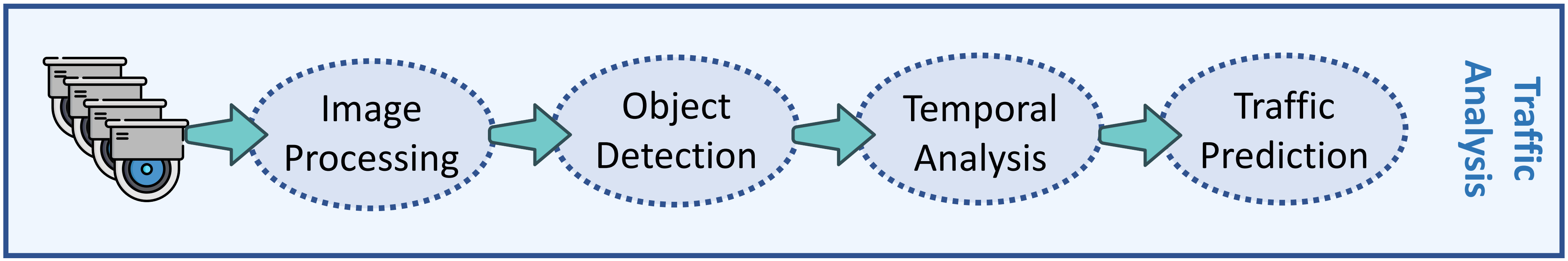
Discussions, Future Work and Conclusions

IoT: The Future Internet

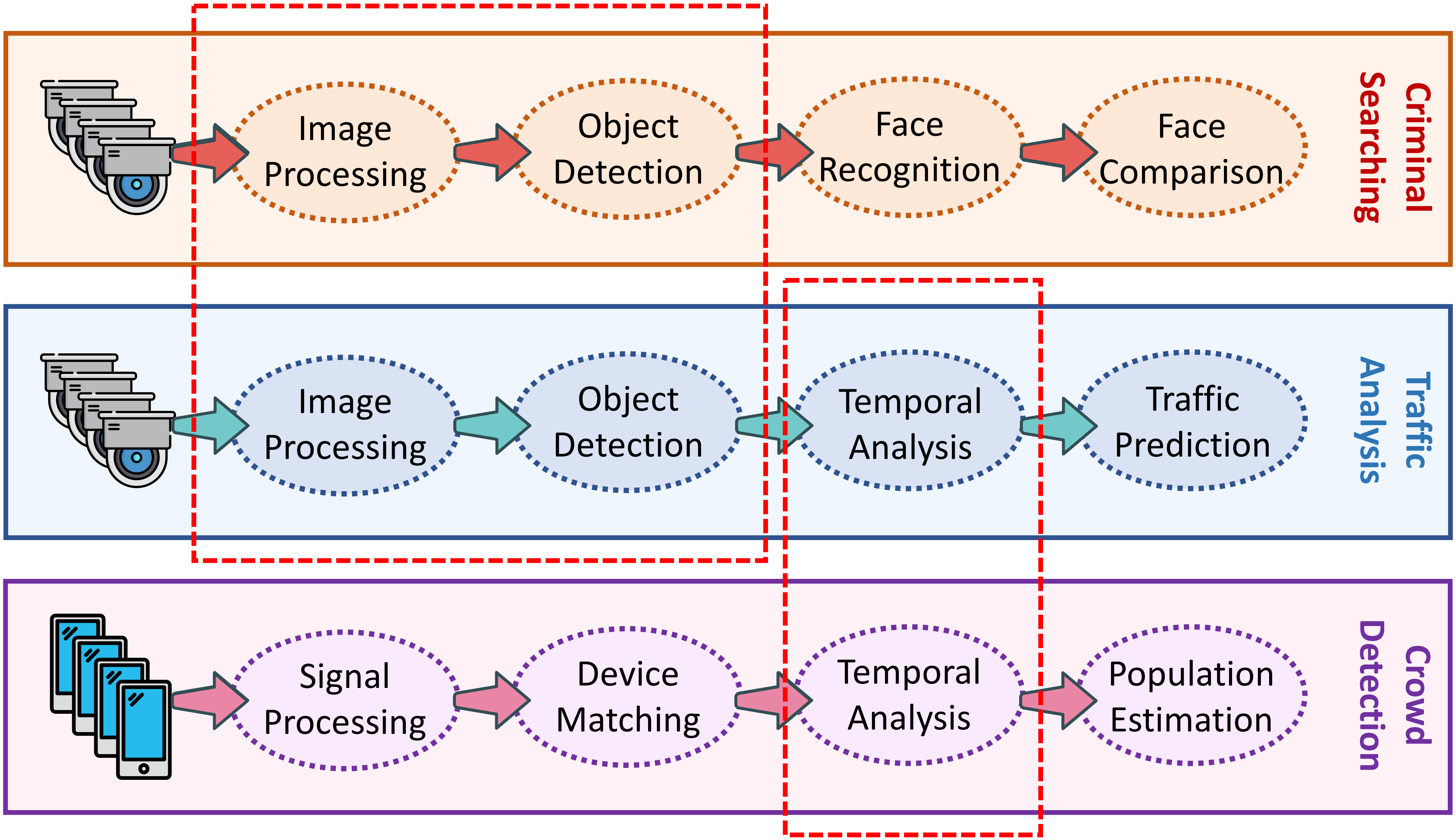
- IoT is the future Internet that connects every aspect of our work and life.



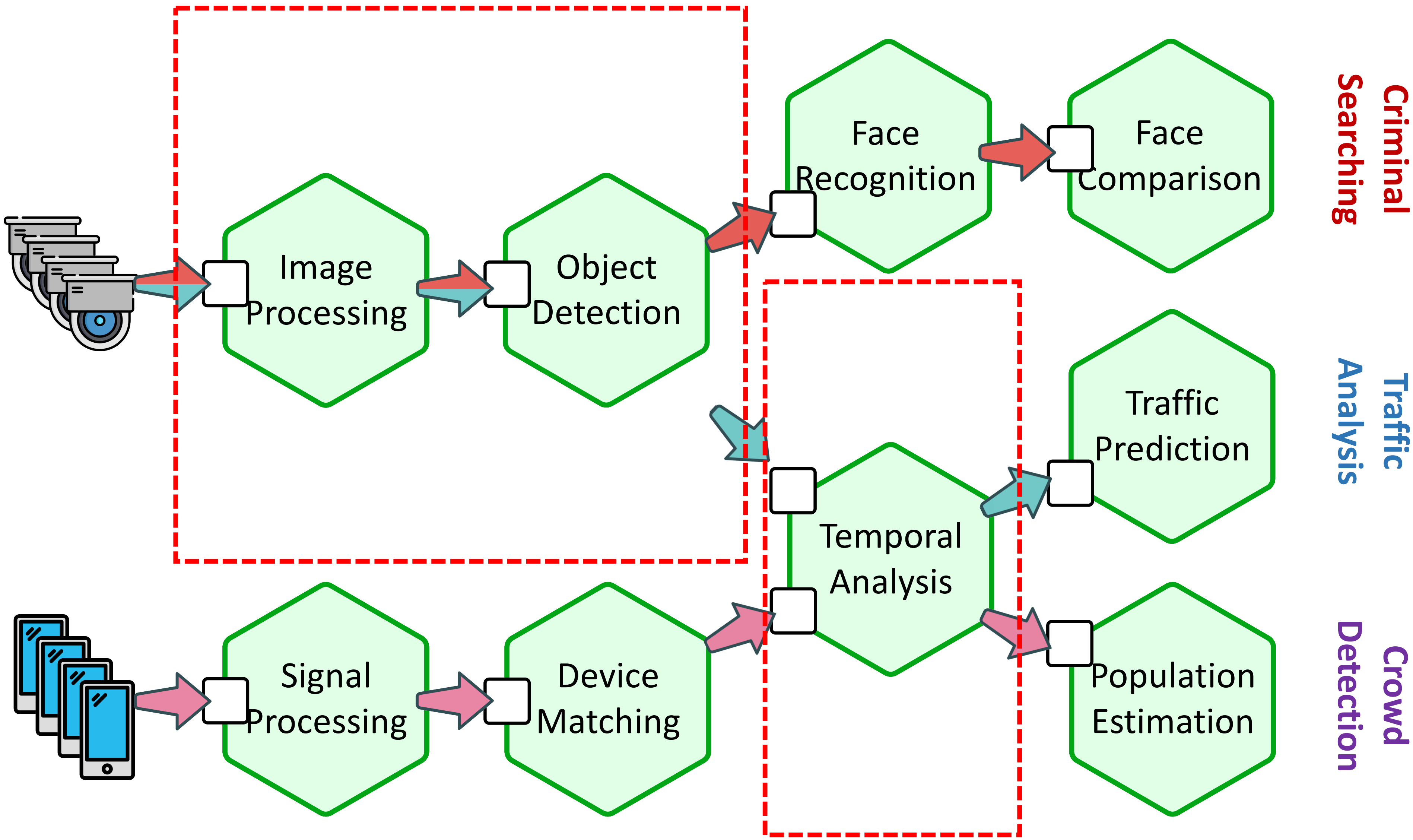
Example IoT Applications



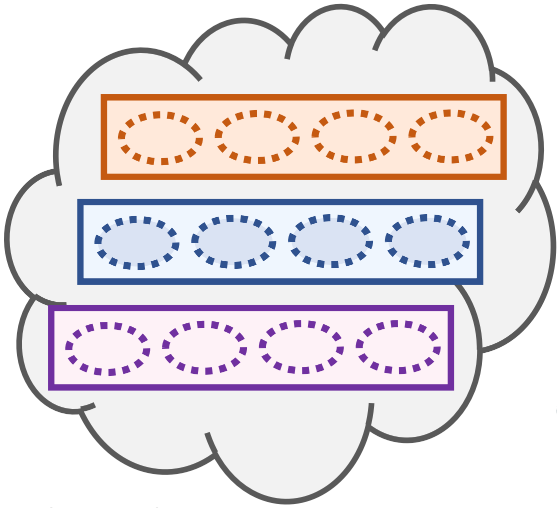
Monolithic Applications



Microservices



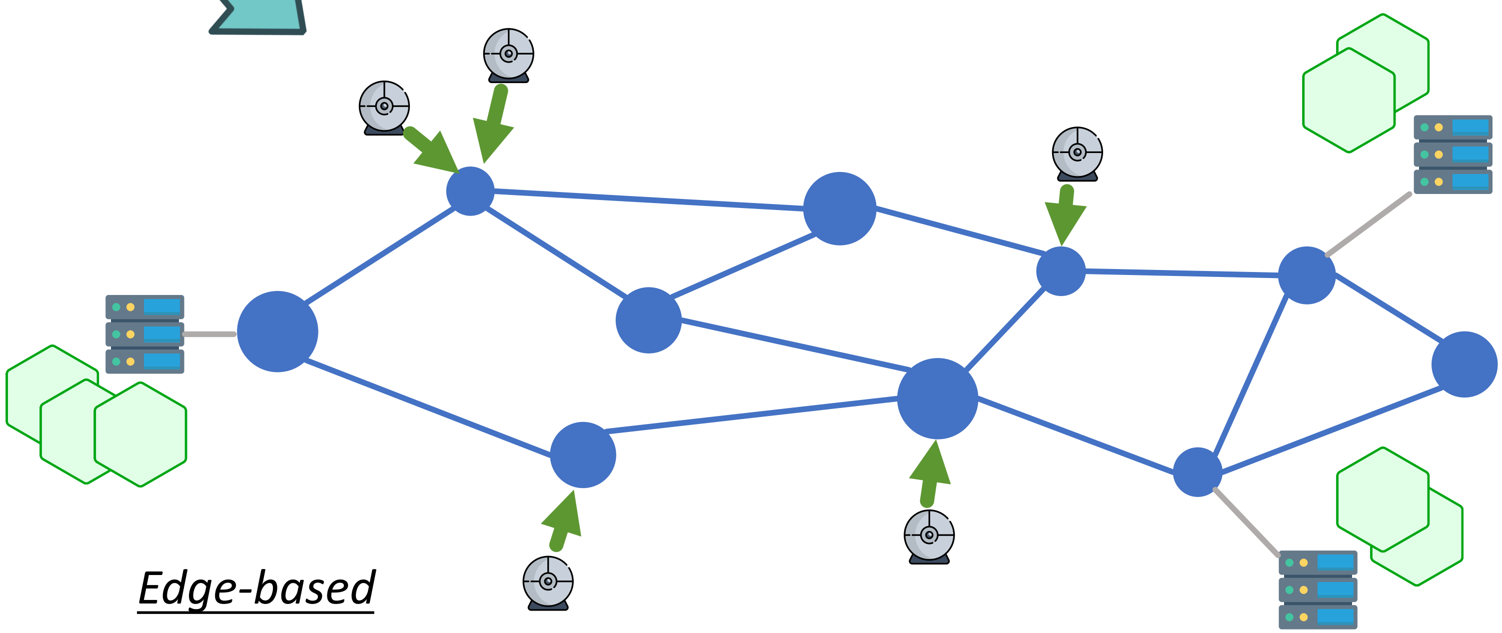
Microservices vs. Edge Computing



Cloud-based

- High delay
- High bw overhead
- Single point of failure
- Security concerns

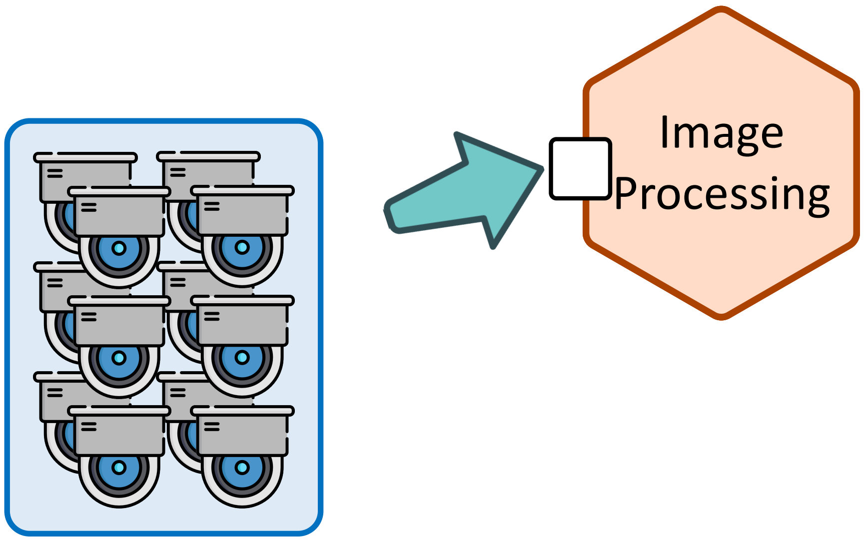
- Responsive
- Low bw overhead
- Distributed
- Secure



Edge-based

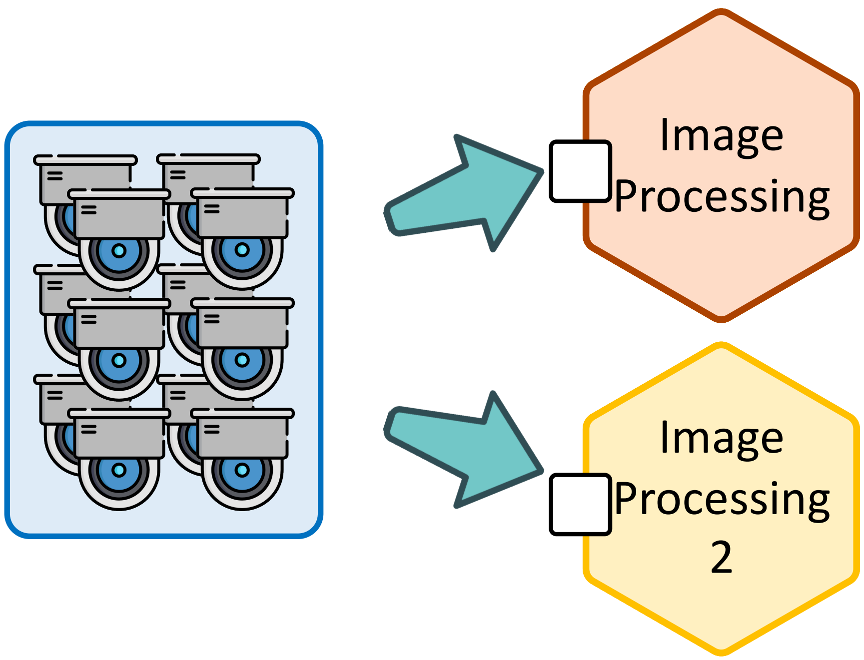
The Microservice Load Balancing Problem

- Edge-based microservices can be easily **saturated**.



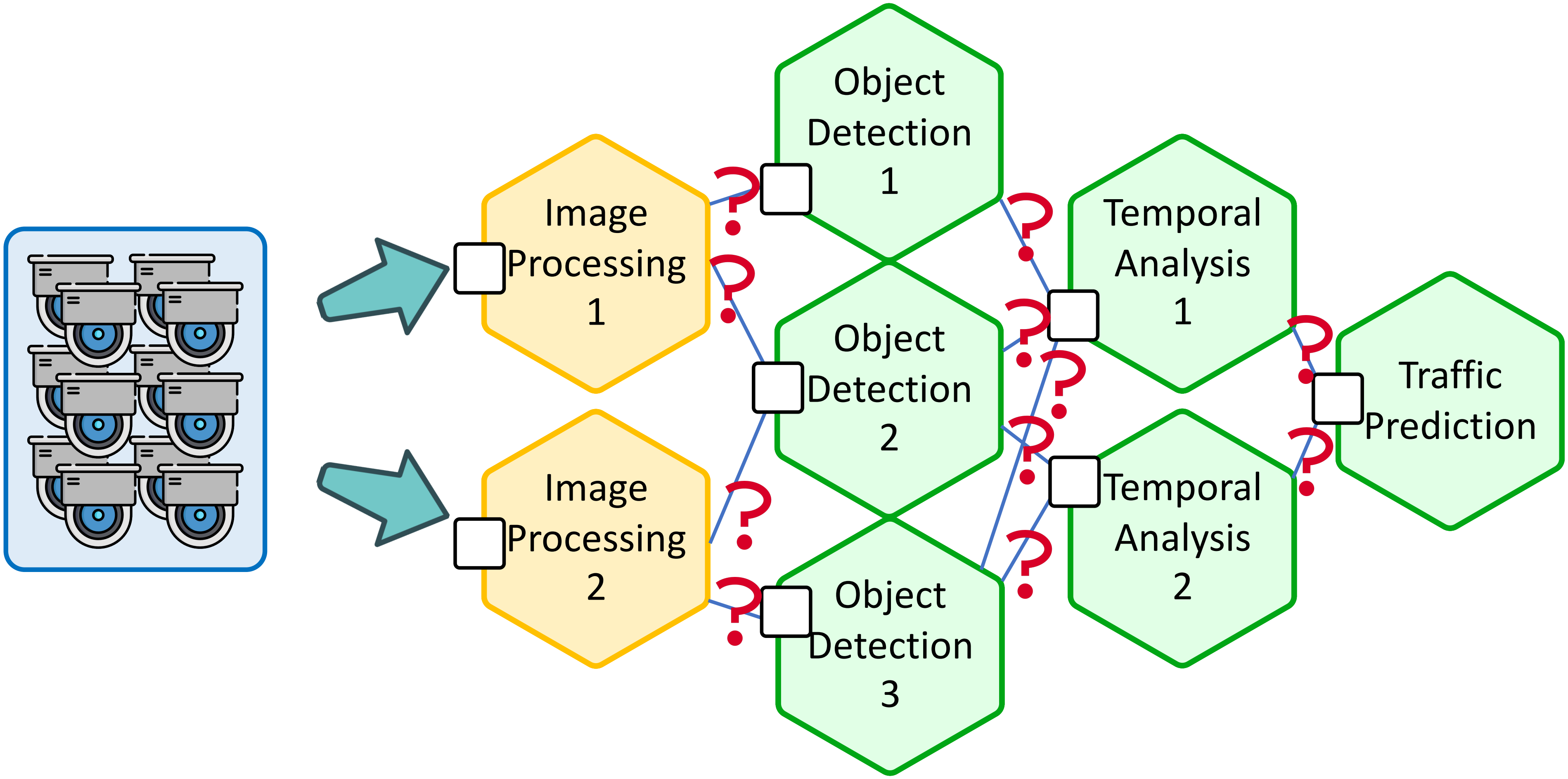
The Microservice Load Balancing Problem

- Edge-based microservices can be easily **saturated**.



The Microservice Load Balancing Problem

- Edge-based microservices can be easily **saturated**.
- **Challenge:** **interdependent** microservices.



Our Approach: Overview

Problem Modeling

1) DAG-based interdependency graph (App-Graph).
2) Compactly modeled infrastructure (Inf-Graph).
3) Flexible application instantiation (Real-Graph).
4) Joint instantiation finding & load allocation.
5) Application QoS requirements.

Algorithmic Results

1) Optimal algorithm for QoS-agnostic problem.
2) NP-hardness for QoS-aware problem.
3) FPTAS for QoS-aware problem.

Next Steps (Future Work)

1) Network-aware load balancing.
2) Reliability and security.
3) Economics-aware microservice composition.

Outlines

Background and Motivation

System Modeling

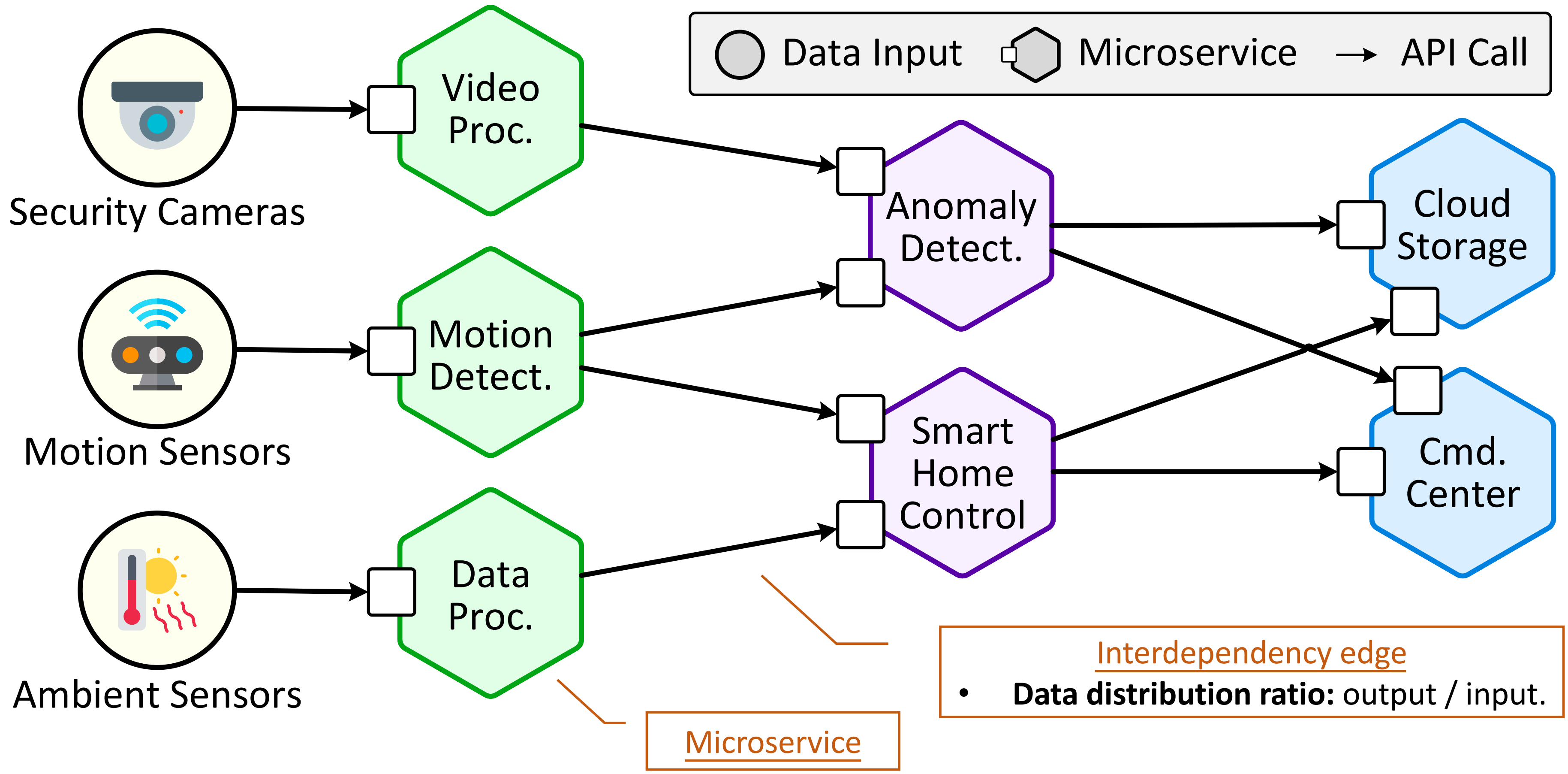
Algorithm Design and Analysis

Performance Evaluation

Discussions, Future Work and Conclusions

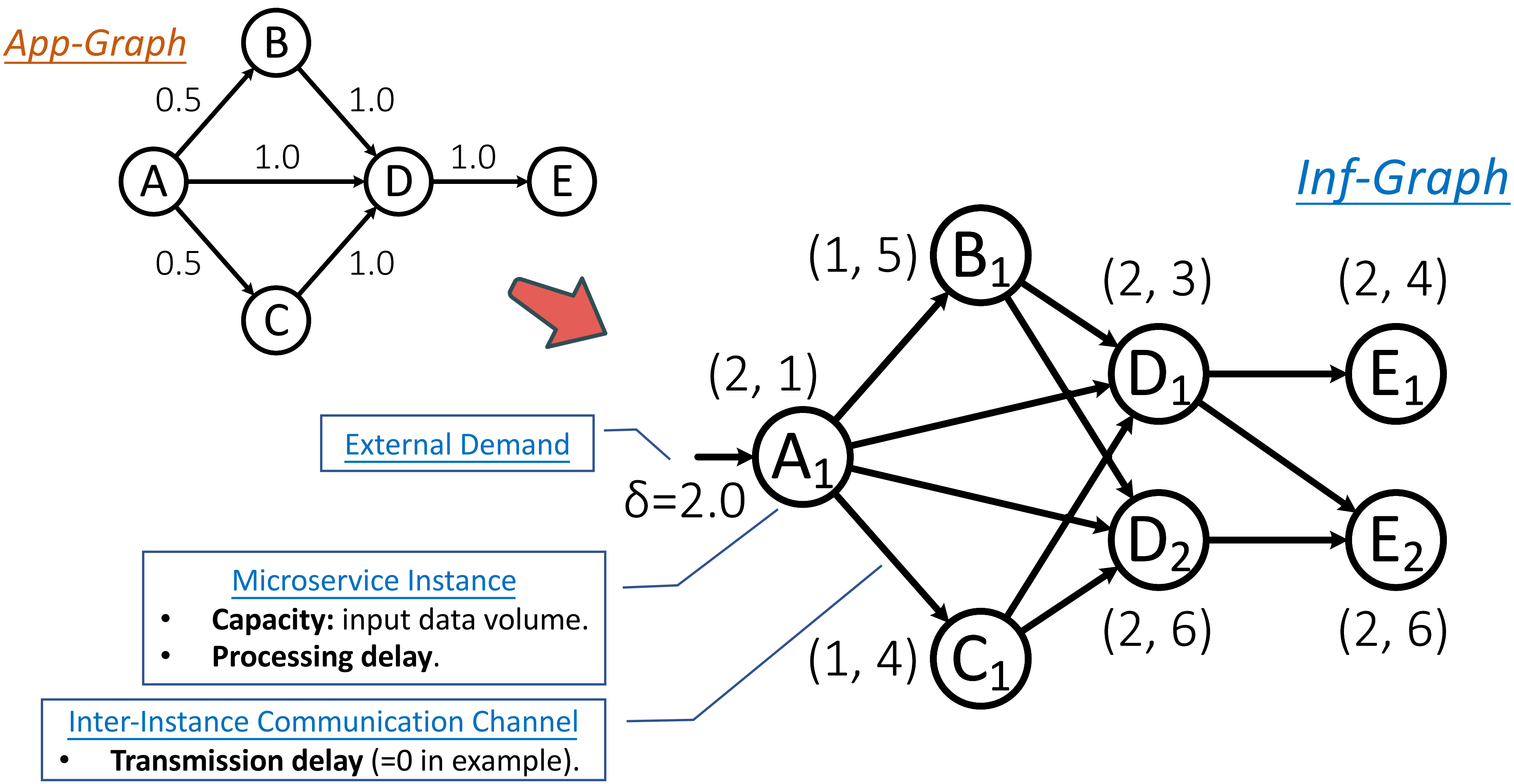
Application with Interdependent Microservices

- General DAG-based application graph (App-Graph).
  - *Captures complex interdependencies, unlike existing line graph-based models.*



IoT Infrastructure in the Application's View

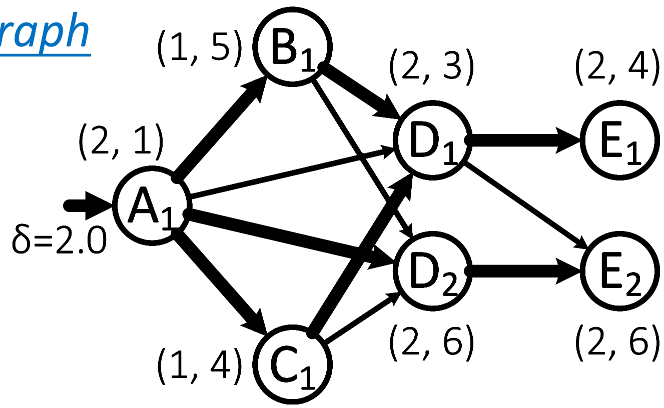
- Inf-Graph: deployed microservice instances & their interactions.



Application Instantiation

- Real-Graph: instantiating the App-Graph in the Inf-Graph.

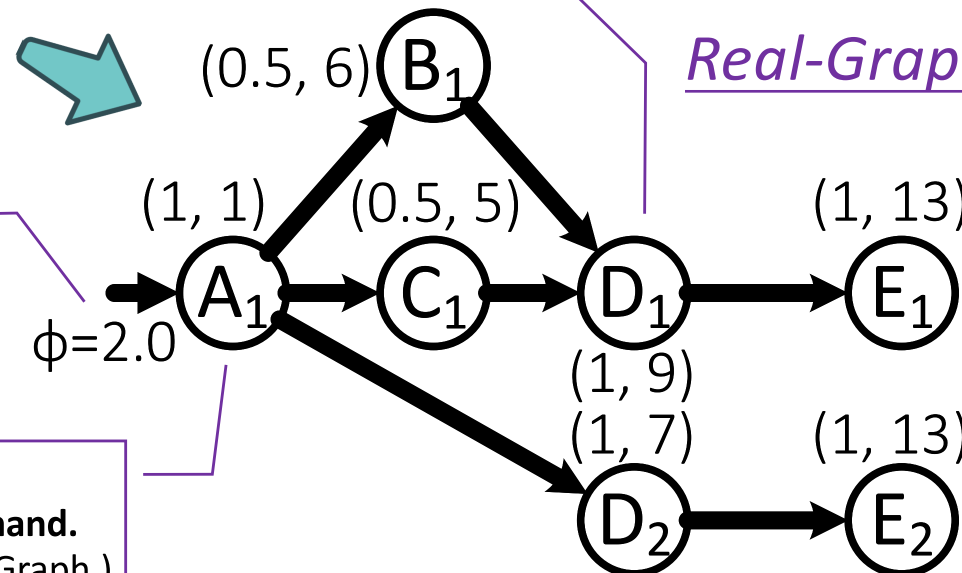
Inf-Graph



Selected Instance

- **Impact ratio:** how much data this instance gets if 1 unit is allocated to the source.
- **Max cumulative delay** from source**.**
- *Both can be computed based on this graph.*

Real-Graph



Source Demand Allocation

Source of Demand

- **A real-graph has a single source of demand.** (There could be multiple sources in Inf-Graph.)

Problem Statement: Overview

- **Inputs:**
  - App-Graph: microservices, interdependencies, data distribution ratios
  - Inf-Graph: instances, communication channels, capacities, and delays

- **Outputs:**
  - A set of Real-Graphs.
  - External demand allocation for each Real-Graph.

- **Constraints:**
  - **Load balancing**: total load <= instance capacity * Ψ.
  - **QoS awareness:** maximum delay <= D.

- **Objective (optimization version):**
  - Minimize <u>maximum delay of all Real-Graphs</u>.

Outlines

Background and Motivation

System Modeling

Algorithm Design and Analysis

Performance Evaluation

Discussions, Future Work and Conclusions

BLB: Basic (QoS-agnostic) Load Balancing

- Without delay constraint, the problem can be formulated as LP.

$$
\begin{aligned}
\text{find} \quad & f : L \mapsto \mathbb{R}^* \\
\text{s.t.} \quad & \delta_n = \delta_n^{\text{ext}} + \sum_{l \in L_{\text{in}}(n)} f(l), \quad \forall n \in N; \\
& \delta_n \leq \Psi \cdot c_n, \quad \forall n \in N; \\
& r_{(v_n, w)} \delta_n = \sum_{l \in L_{\text{out}}(n,w)} f(l), \quad \forall n, w \in V_{\text{out}}(v_n).
\end{aligned}
$$

Variables: Per-link demand allocation function.

Demand per node = external + flow-in.

Capacity (load balancing) per node.

Flow conservation: sum flow towards all instances of a downstream microservice *w* = input data * data distribution ratio of *w*.

BLB: Basic (QoS-agnostic) Load Balancing

- Without delay constraint, the problem can be formulated as LP.

$$
\begin{aligned}
\text{find} \quad & f : L \mapsto \mathbb{R}^* \\
\text{s.t.} \quad & \delta_n = \delta_n^{\text{ext}} + \sum_{l \in L_{\text{in}}(n)} f(l), \quad \forall n \in N; \\
& \delta_n \leq \Psi \cdot c_n, \quad \forall n \in N; \\
& r_{(v_n, w)} \delta_n = \sum_{l \in L_{\text{out}}(n,w)} f(l), \quad \forall n, w \in V_{\text{out}}(v_n).
\end{aligned}
$$

Variables: Per-link demand allocation function.

Demand per node = external + flow-in.

Theorem 1:

BLB is optimally solvable in polynomial time.

ARIZONA STATE UNIVERSITY

Real-Graph Decomposition Theorem

Theorem 2:
Every demand allocation function *f* so defined can be decomposed into at most $|N| + |L|$ real-graphs with positive source demands.

- Why do we need such a theorem?

1. Transform any solution of BLB into a set of implementable real-graphs.
2. Define QoS of a load balancing plan (max delay of all real-graphs).

QLB: QoS-aware Load Balancing

Theorem 3:

QLB (optimization version) is NP-hard.

- **Fully Polynomial-Time Approximation Scheme (FPTAS)** can achieve the best trade-off between time and accuracy
  - Approximation ratio: $(1+\epsilon)$ – For maximization problem
  - Time complexity: $O(\text{poly}(1/\epsilon) \times \text{poly}(\text{input}))$
  - In practice, one can arbitrarily tune ϵ to get best accuracy within time limit.

Theorems 4&5:

QLB (optimization version) admits an FPTAS.

A Brief Overview of Our FPTAS

- Idea:
  - Pseudo-polynomial time algorithm:
    - Expand Inf-Graph into a delay-layered graph.
    - Run BLB LP on the expanded graph.
  - Discretization via approximate testing:
    - Find delay lower & upper bounds (UB, LB) s.t. UB <= poly(input) * LB.
    - Discretize delay values based on (UB, LB).
    - Run pseudo-polynomial time algorithm.
    - Refine (UB, LB) based on output.

$$O\left(\frac{1}{\epsilon^4}|L|^3|N|^8\mathbb{L}\log\frac{|N|}{\epsilon}+|L|^3\mathbb{L}\log|N|\right)$$

  - Efficiency enhancement:
    - Approximate testing to shrink initial bound s.t. UB <= constant * LB.

$$O\left(\frac{1}{\epsilon^4}|L|^3|N|^4\mathbb{L}\log\frac{|N|}{\epsilon}+|L|^3|N|^4\mathbb{L}\log\log|N|\right)$$

Outlines

Background and Motivation

System Modeling

Algorithm Design and Analysis

Performance Evaluation

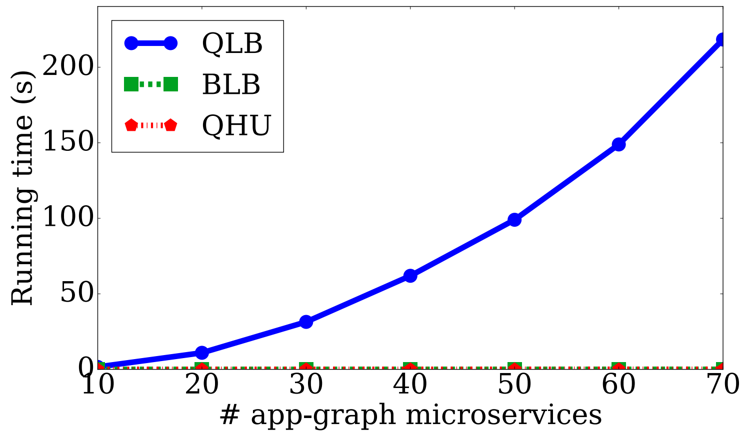
Discussions, Future Work and Conclusions

Simulation Settings

- Simulated network scenarios:
  - App-Graph:
    - 5-layered applications, layer-1 being the source layer
    - 10-70 microservices: 10% in layer-1, uniformly distributed in other layers
    - 4 in-going edges per microservice in layers 2-5
    - Data distribution ratio: uniformly generated
  - Inf-Graph:
    - 1 instance per microservice in source layer, 5-15 in others
    - Linking probability (between interdependent instances): 0.3
    - Source demands: 100-900 units
    - Node capacities: 10-90 units
    - Node/Link delays: 0-500/1000 ms
  - Load balancing goal: optimal load under BLB, or 2 x optimal load under BLB
  - Approximation parameter: ϵ=0.5

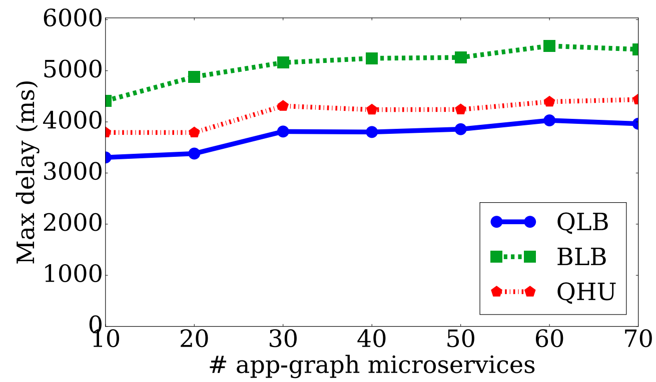
- Comparisons:
  - QLB
  - BLB
  - QHU: QoS-aware heuristic, solving BLB minimizing demand-weighted delay

Comparison Results



Running time

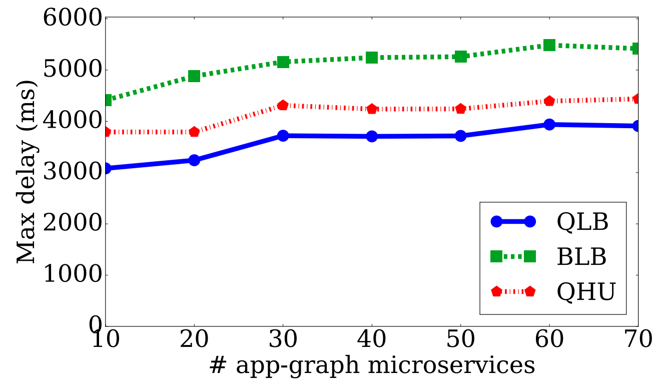
Polynomially increased



$\Psi = \psi^*$

Max delay (QoS)

QLB has improved delay over the other solutions.



$\Psi = 2\psi^*$

Outlines

Background and Motivation

System Modeling

Algorithm Design and Analysis

Performance Evaluation

Discussions, Future Work and Conclusions

Other Perspectives and Beyond

- So far, we've talked about
  - Basic model: DAG-based apps, Real-Graphs
  - Processing capacities and delays
  - Network delays

  Computing Perspective

- What we didn't consider in this work
  - Network topology
  - Network capacities & congestion
  - Routing

  Networking Perspective

  - Reliability: microservice instance failures

  Security Perspective

  - Incentives, pricing
  - Payment methods

  Economics Perspective

- A **unified approach** is **still** in need for high-performance IoT.

Our Conclusions

- Load Balancing for Interdependent IoT Microservices
  - DAG model for applications: App-Graph and Inf-Graph
  - Application realization with Real-Graph abstraction
  - System-wide load balancing with QoS (delay) constraints

- Algorithmic solutions
  - Optimal solution for QoS-agnostic problem
  - FPTAS for (NP-hard) QoS-aware problem

- Future directions
  - Unified framework for IoT performance optimization

Thank you very much!

Q&A?

NP-Hardness Proof