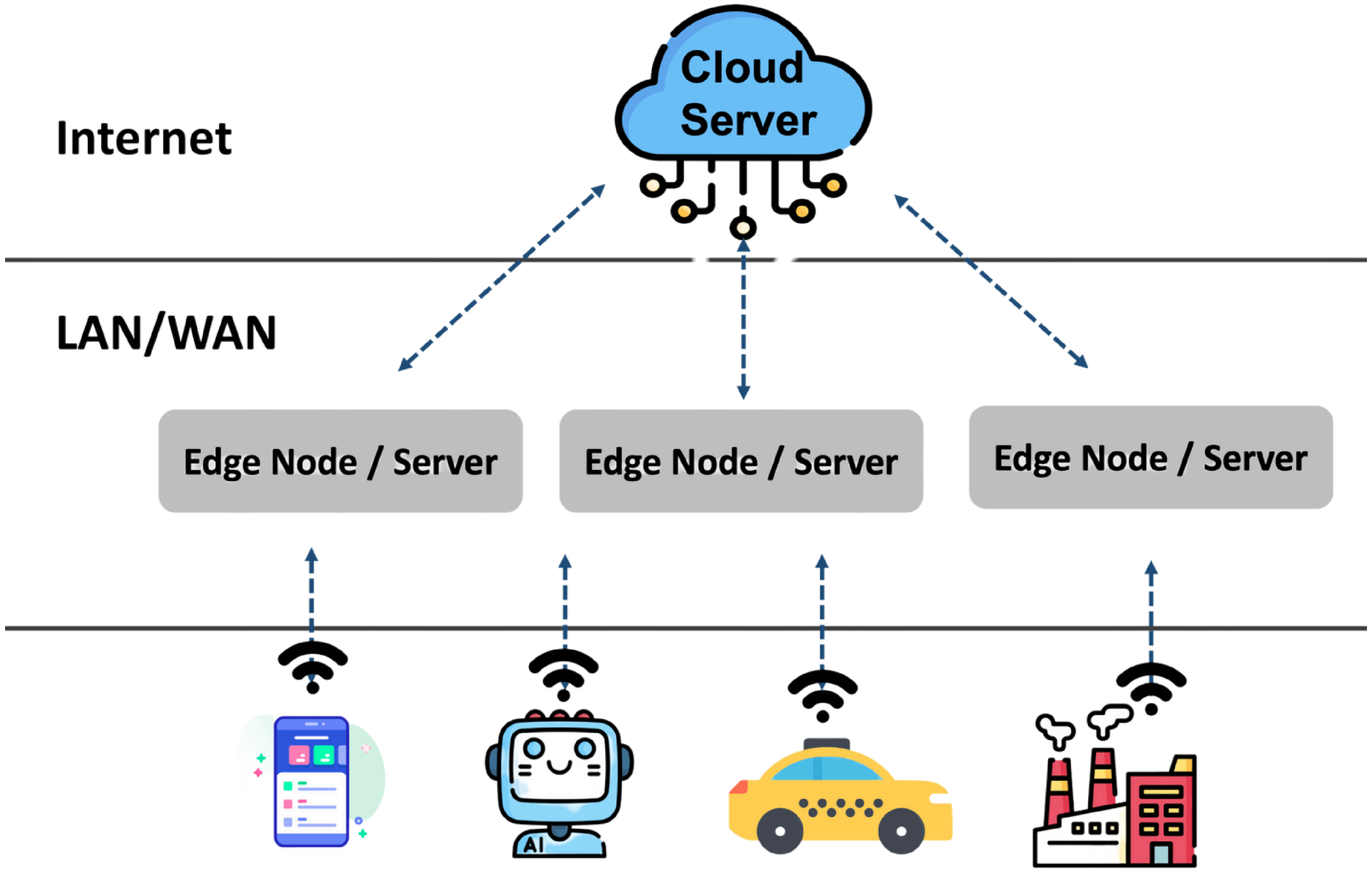
TrustMesh

A Blockchain-Enabled Trusted Distributed Computing Framework for Open Heterogeneous IoT Environments

Murtaza Rangwala and Rajkumar Buyya

Cloud Computing and Distributed Systems (CLOUDS) Lab
Department of Computing and Information Systems
The University of Melbourne

Introduction



Primitive View of an Edge Computing Network

Cloud Layer

- Big data processing
- Data warehousing

Edge Layer

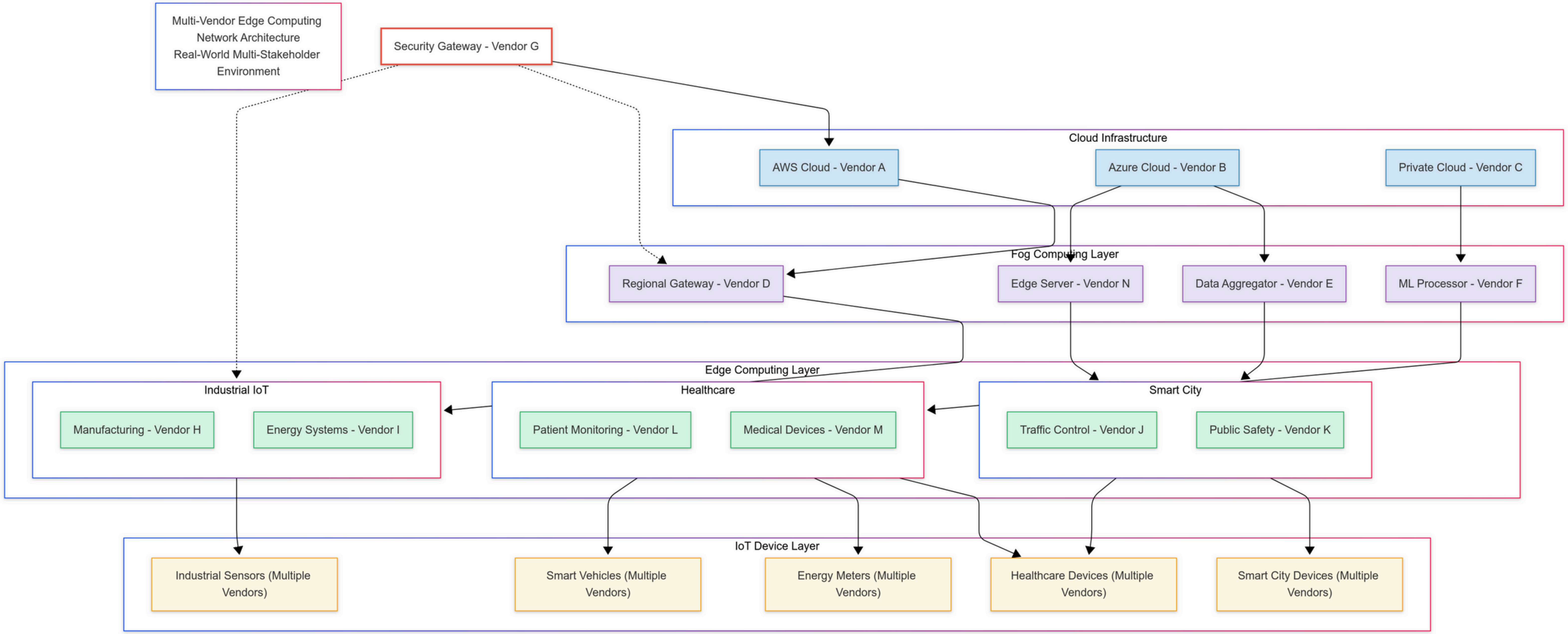
- Local network
- Data Processing & Reduction
- Data Caching & Buffering
- Control Response
- Virtualization

Device Layer

- Sensors & Controllers

Introduction

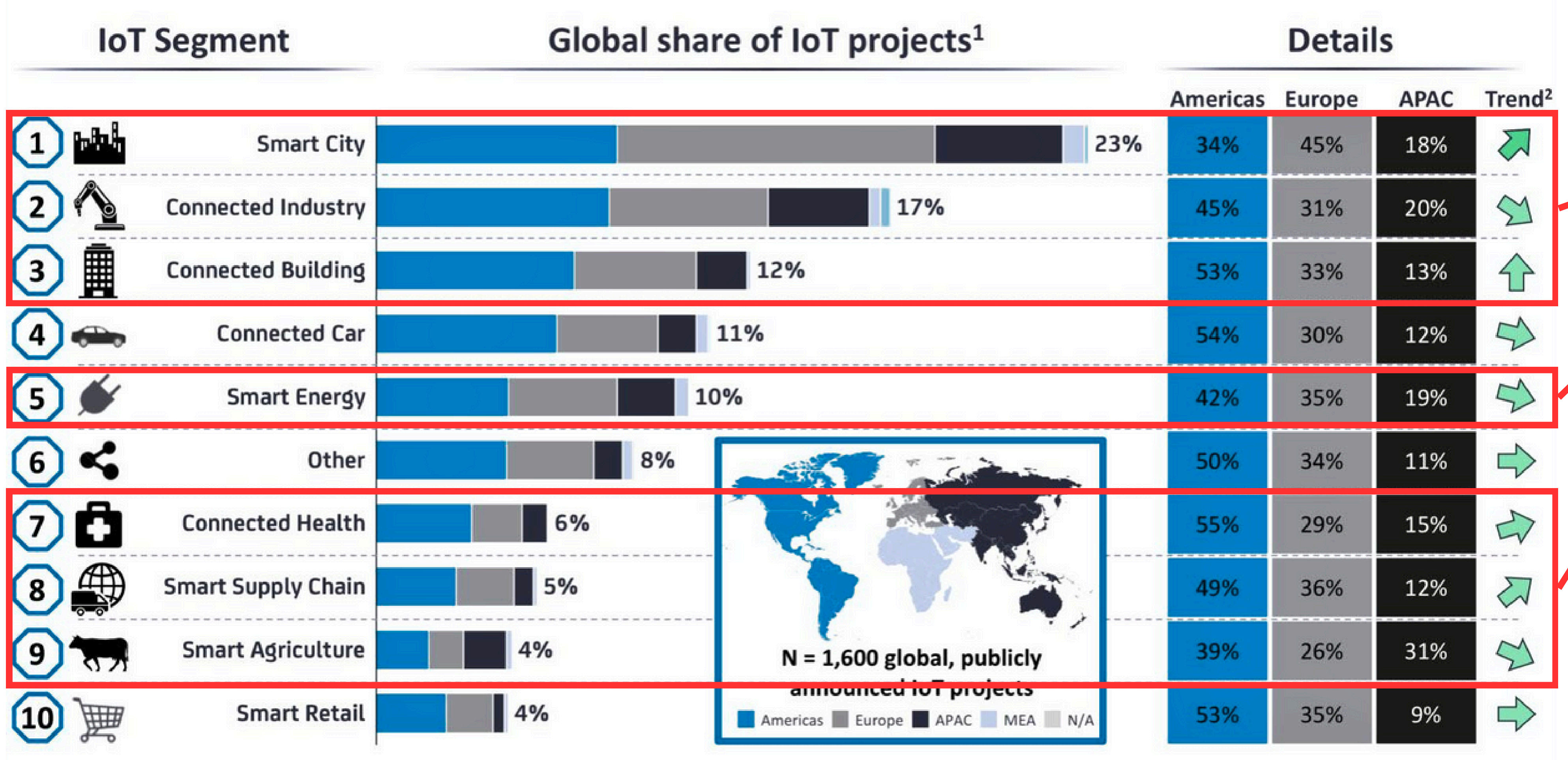
Real-world multi-stakeholder environments are much more complex!



Challenges

THE UNIVERSITY OF MELBOURNE

- Globally, IoT connections are growing at a CAGR of 16%
- Revenue from these connections is growing at a CAGR of 14%



IoT Segment	Global share of IoT projects[1]	Americas	Europe	APAC	Trend[2]
1 Smart City	23%	34%	45%	18%	↗
2 Connected Industry	17%	45%	31%	20%	↘
3 Connected Building	12%	53%	33%	13%	↑
4 Connected Car	11%	54%	30%	12%	→
5 Smart Energy	10%	42%	35%	19%	→
6 Other	8%	50%	34%	11%	→
7 Connected Health	6%	55%	29%	15%	→
8 Smart Supply Chain	5%	49%	36%	12%	↗
9 Smart Agriculture	4%	39%	26%	31%	↘
10 Smart Retail	4%	53%	35%	9%	→

1.Based on 1,600 publicly known enterprise IoT projects (Not including consumer IoT projects e.g., Wearables, Smart Home). 2.Trend based on comparison with % of projects in the 2016 IoT Analytics Enterprise IoT Projects List. A downward arrow means the relative share of all projects has declined, not the overall number of projects 3. Not including Consumer Smart Home Solutions. **Source:** IoT Analytics 2018 Global overview of 1,600 enterprise IoT use cases (Jan 2018)

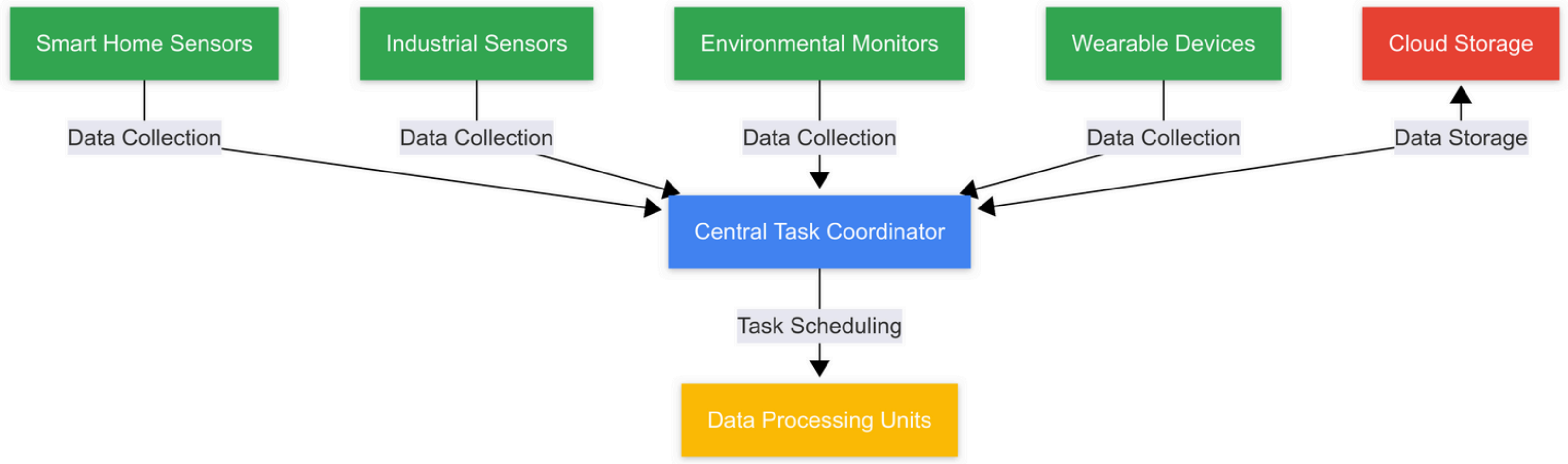
Multi-Stakeholder Environments

- **Data Integrity**
- **Lack of Centralized Trust**
- **Stringent Audit Requirements**
- **Device Heterogeneity**

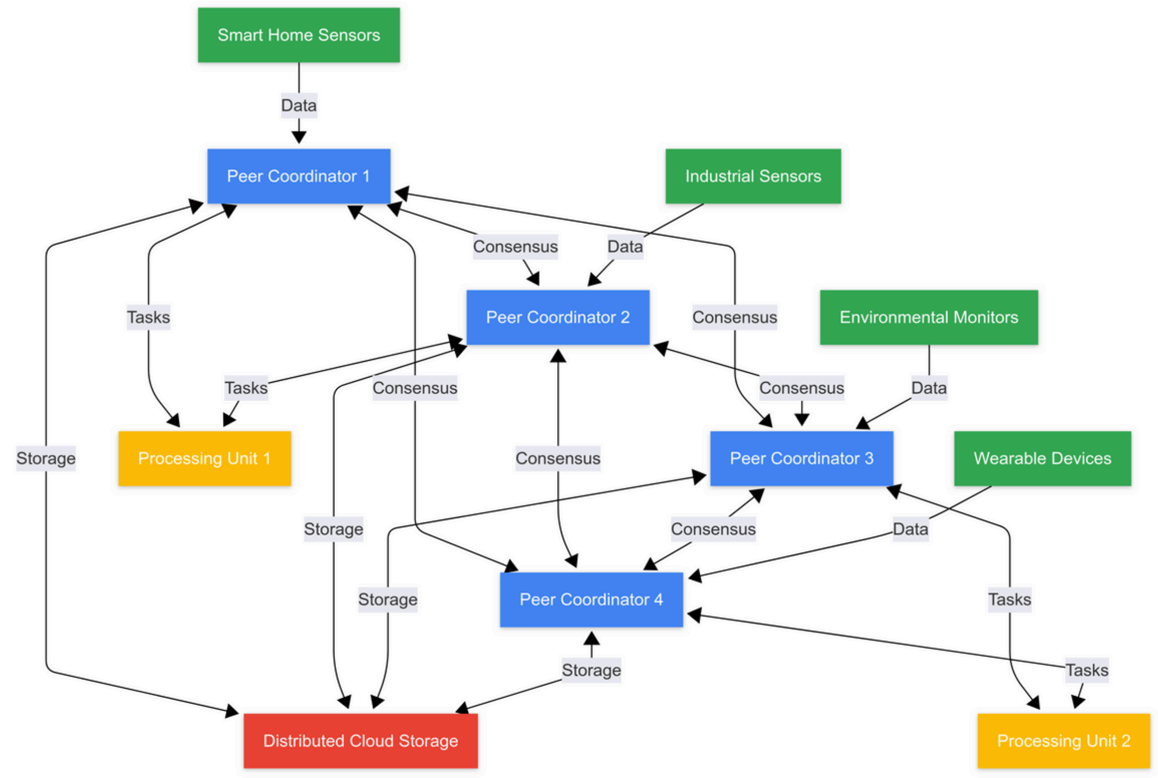
Existing Work

Existing distributed computing frameworks broadly fall under two distinct design philosophies

Central Coordinator Model



Distributed Consensus Model



Existing Work

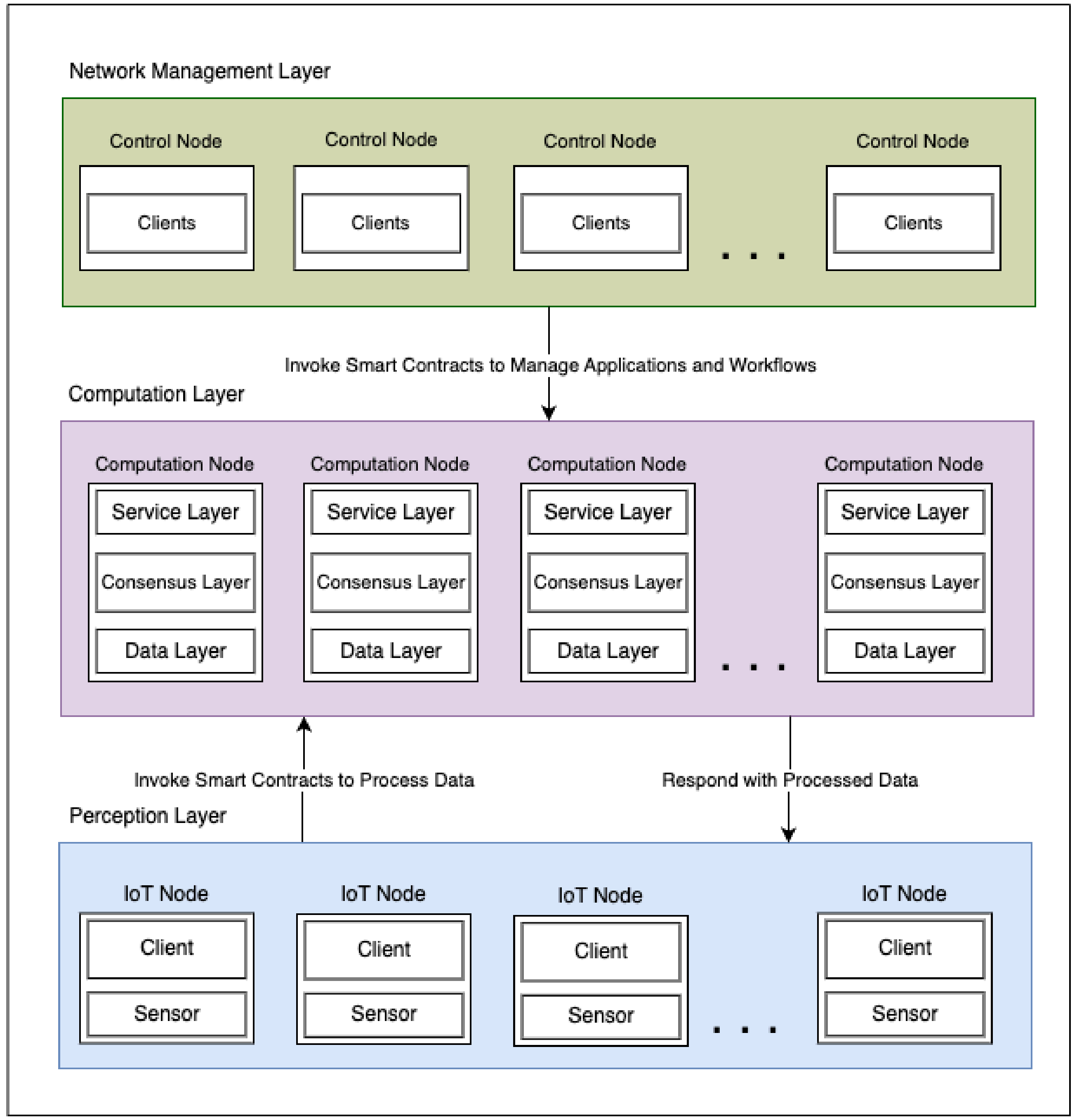
Existing distributed computing frameworks broadly fall under two distinct design philosophies

Central Coordinator Model	Distributed Consensus Model
Allows the use of flexible, non-deterministic scheduling approaches such as **machine learning solutions** and other meta-heuristics	In most cases, **fault tolerant** with minimal reliance on individual entities
Introduces **single point(s) of failure** making the network susceptible to malicious attacks	**Less flexibility** with selection of scheduling approach since consensus requires a deterministic outcome.

TrustMesh: Bridging the Gap

Allows the use of flexible, non-deterministic scheduling approaches such as **machine learning solutions** and other meta-heuristics

Byzantine fault tolerant with minimal reliance on individual entities



Three-Layer Architecture

Network Management Layer: Handles system setup and configuration

Computation Layer: Runs the blockchain network and processes data

Perception Layer: Interfaces with IoT devices

Key Innovations

- Supports non-deterministic scheduling while maintaining Byzantine fault tolerance using a novel **multi-phase commit protocol**
- Maintains **immutable audit trail** and implements **secure data handling** through blockchain and smart contracts
- Capable of handling **diverse IoT devices** efficiently

Network Management Layer

Control Nodes

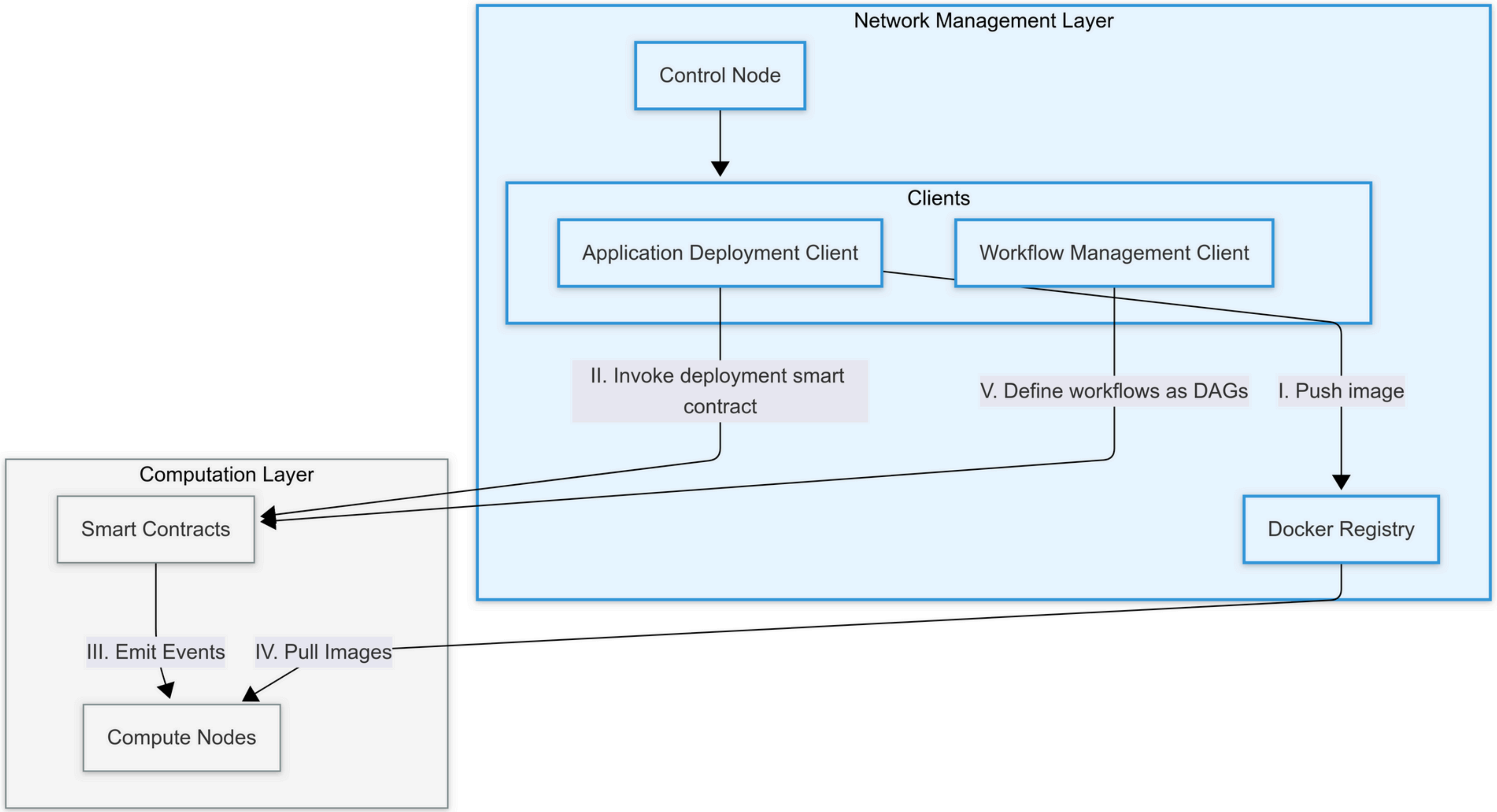
- Serves as access point for administrators
- Manages applications and workflows without direct data processing
- Not a critical point of failure for ongoing operations

Application Deployment

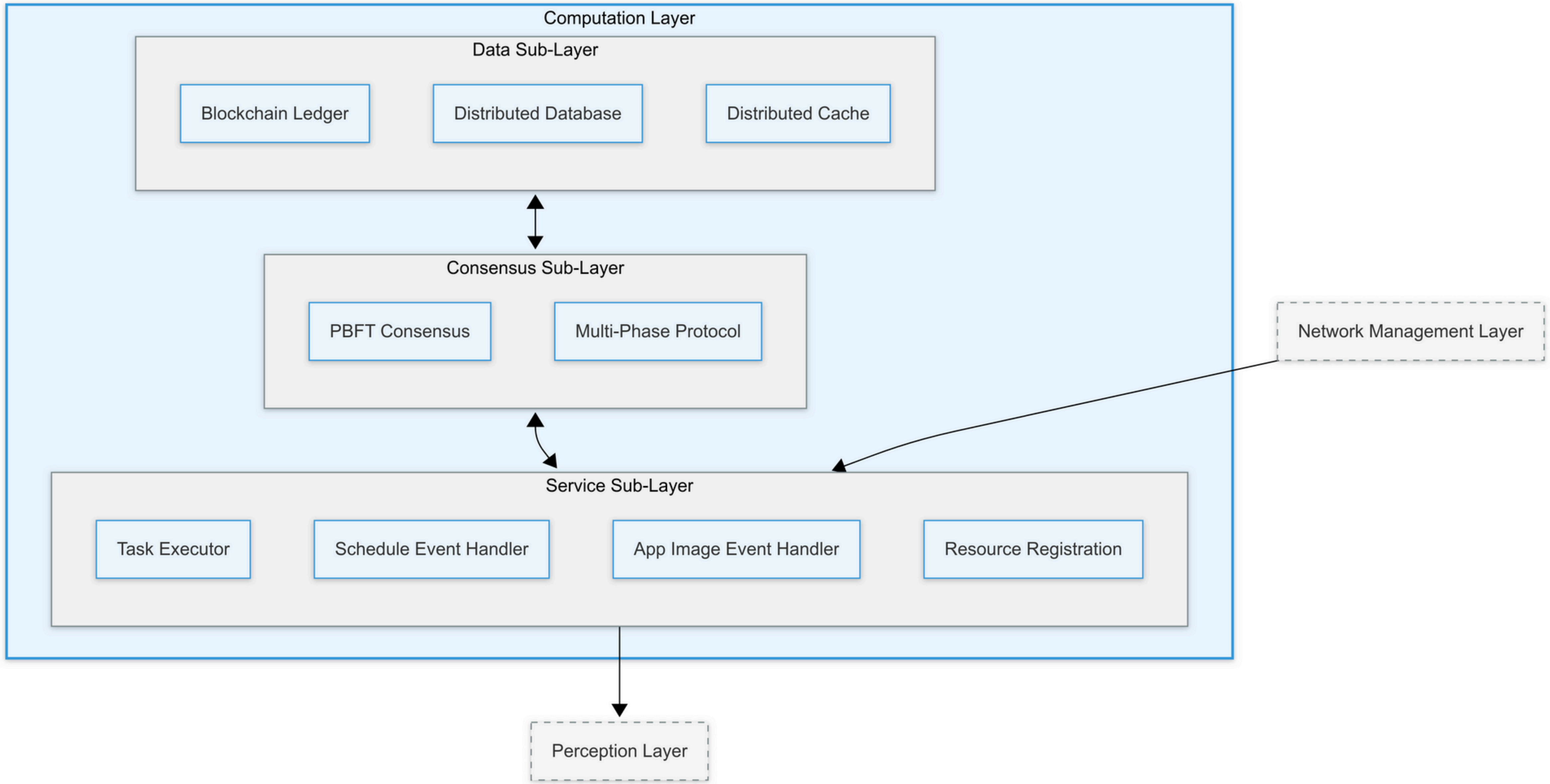
- Images pushed to registry, then smart contracts invoked
- Event-driven architecture ensures consistent application deployment

Workflow Management

- Defines complex processing pipelines as Directed Acyclic Graphs (DAGs)
- Relationships between applications are immutable once stored



Computation Layer



Data Sub-Layer

- **Blockchain Ledger:** Provides an immutable audit trail for all operations
- **Distributed Database:** Handles persistent storage with multi-master architecture
- **Distributed Cache:** Facilitates temporary storage and inter-node communication

Consensus Sub-Layer

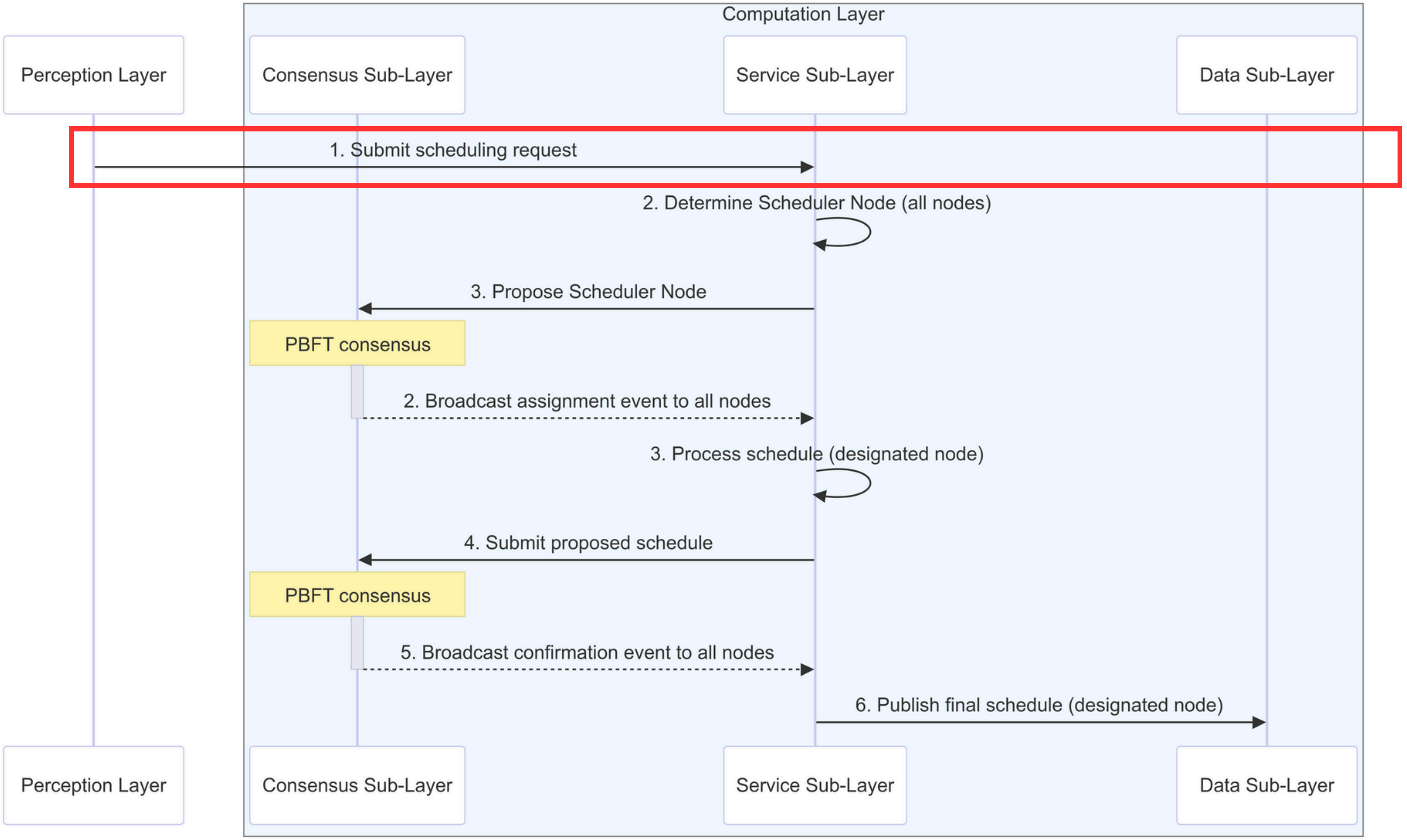
- **PBFT Consensus:** Provides Byzantine fault tolerance for standard operations
- **Multi-Phase Protocol:** Enables non-deterministic scheduling decisions

Service Sub-Layer

- **Task Executor:** Orchestrates workflow execution according to predefined specifications
- **Schedule Event Handler:** Oversees schedule generation within the computation layer
- **App Image Event Handler:** Processes application management events
- **Resource Registration:** Monitors compute node resources at configured intervals

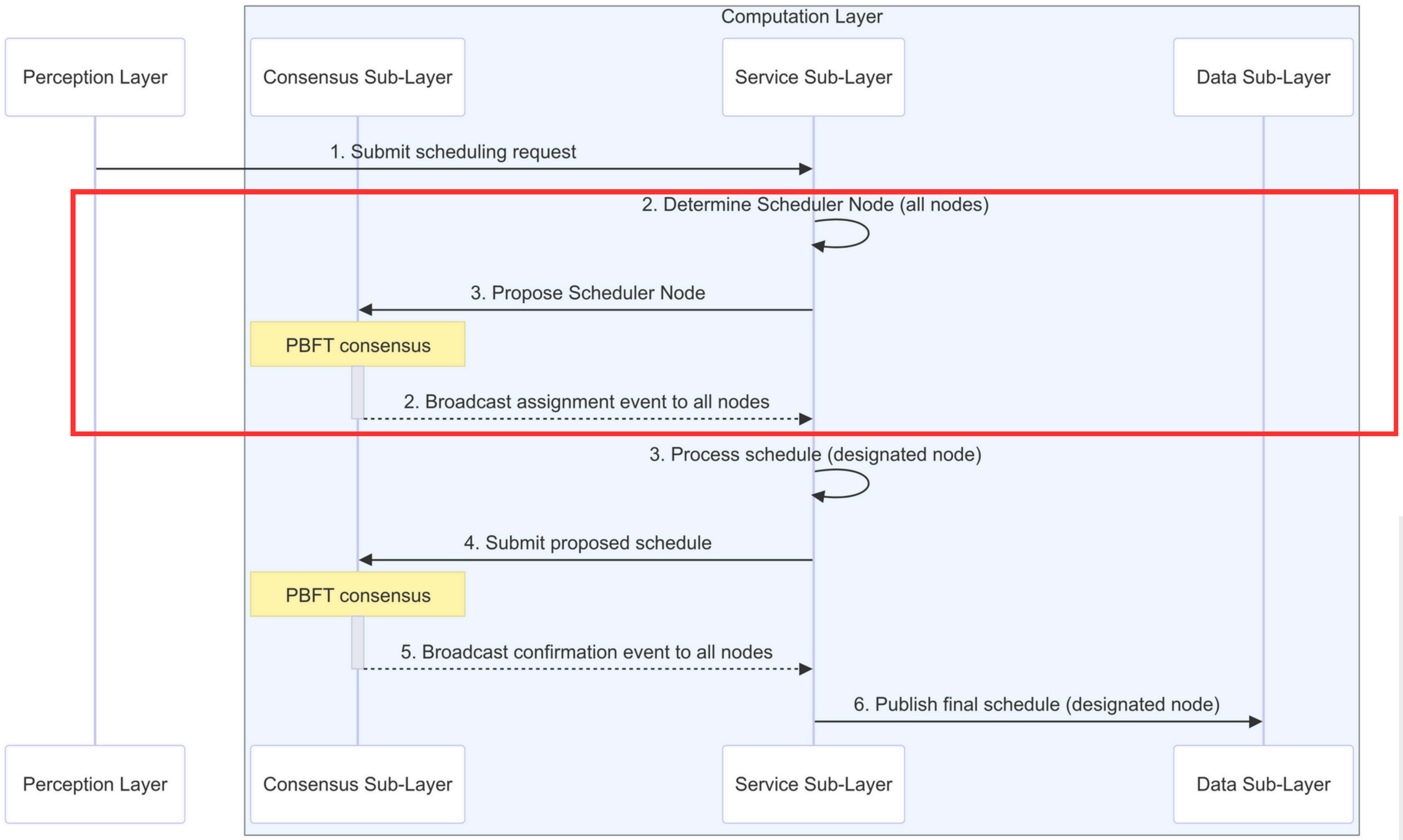
Multi-Phase Commit Protocol

THE UNIVERSITY OF MELBOURNE



IoT Node submits an intent to send data for processing.

Multi-Phase Commit Protocol

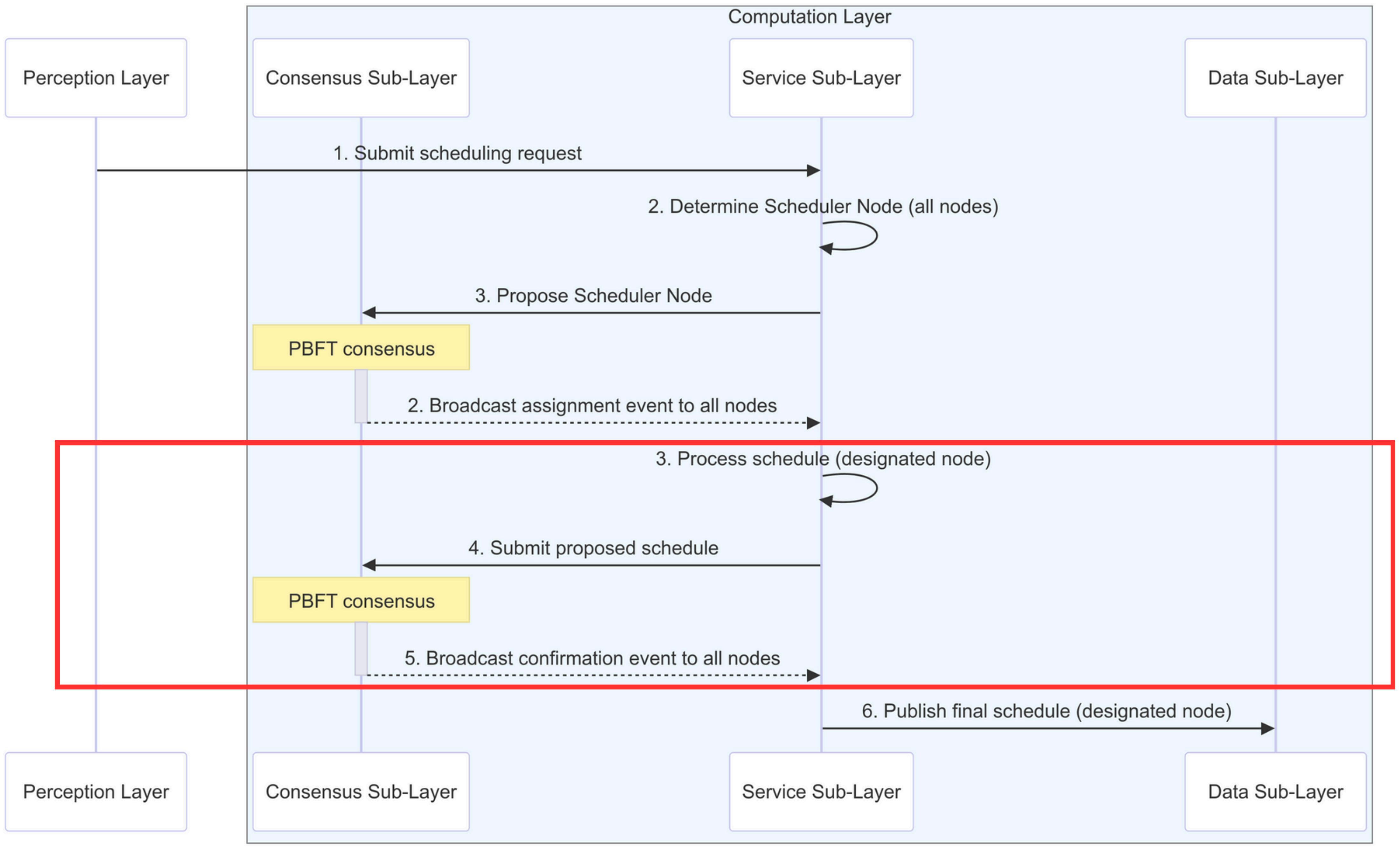


All the compute nodes in the network use a deterministic algorithm to assign the scheduling responsibility to one of the blockchain nodes using PBFT consensus.

$$f(N,R) = n^* \text{ where } n^* = \max_{n \in N}(w_c \cdot c_n + w_m \cdot m_n)$$

with c_n and m_n representing available CPU and memory resources for node n, and w_c, w_m being weighting factors.

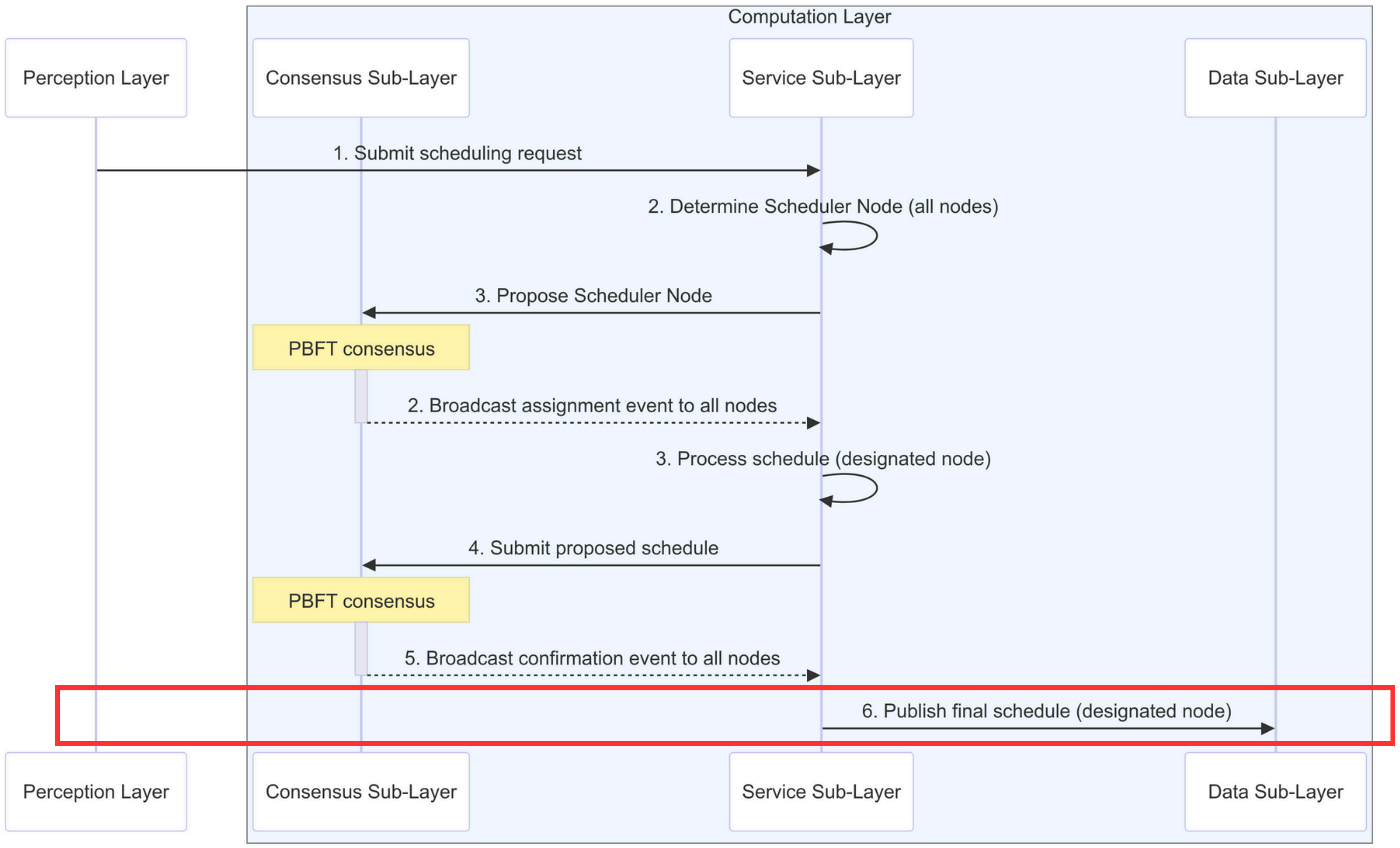
Multi-Phase Commit Protocol



The newly designated scheduler uses the available resource data and a flexible (and pluggable) algorithm to decide on the data processing schedule. All the computes nodes achieve consensus on the generated schedule using PBFT.

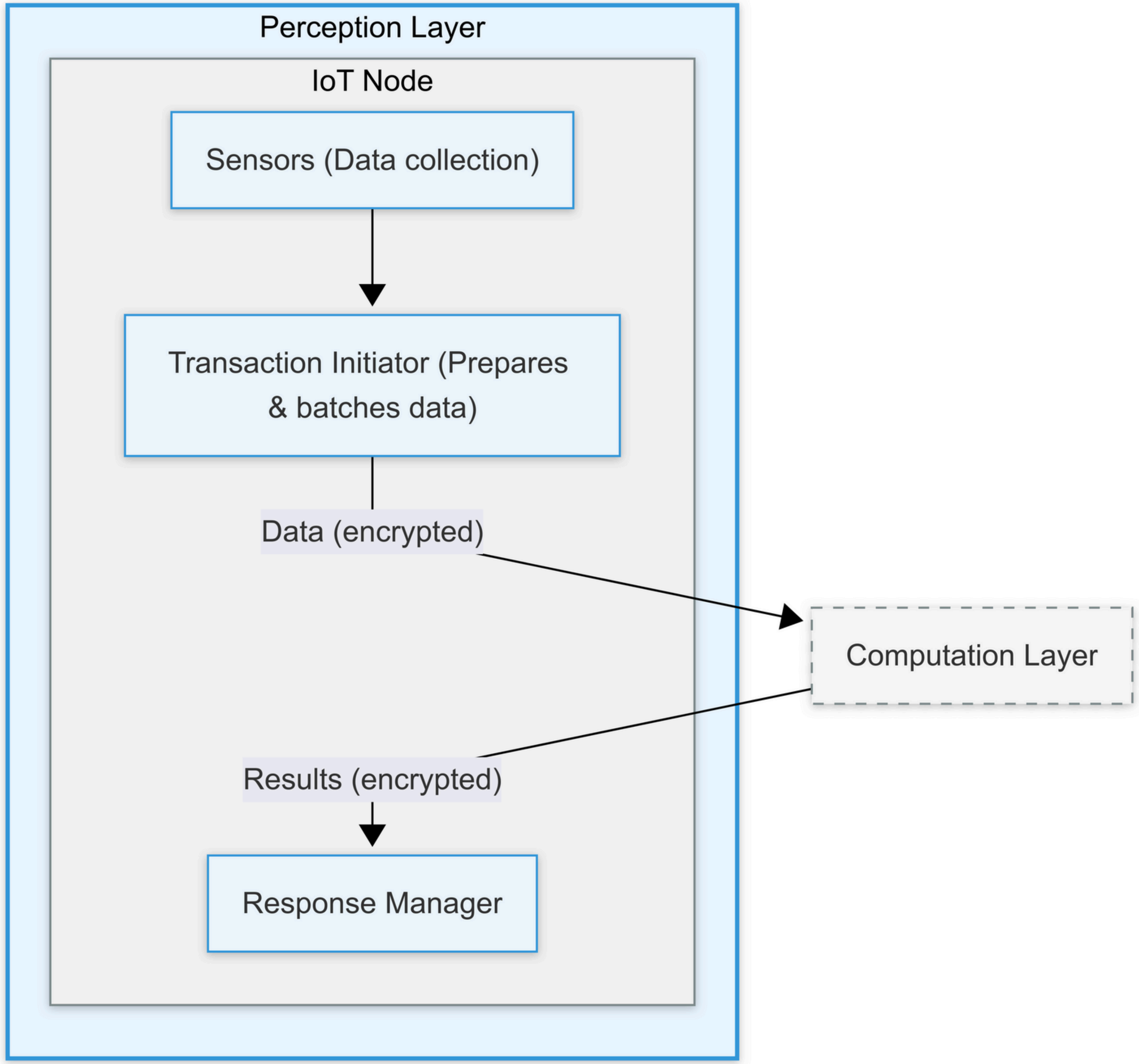
We use a Least-Connected Dynamic Weighted Round Robin (LCDWRR) approach for experimentation and demonstration purposes

Multi-Phase Commit Protocol



Once consensus is achieved, the schedule is stored in the blockchain and the IoT node sends the data to be processed to the assigned node.

Perception Layer



Zero-Process Baseline Approach

- Minimalist design optimized for resource-constrained IoT devices
- No running processes by default upon deployment

Helper Components as Libraries

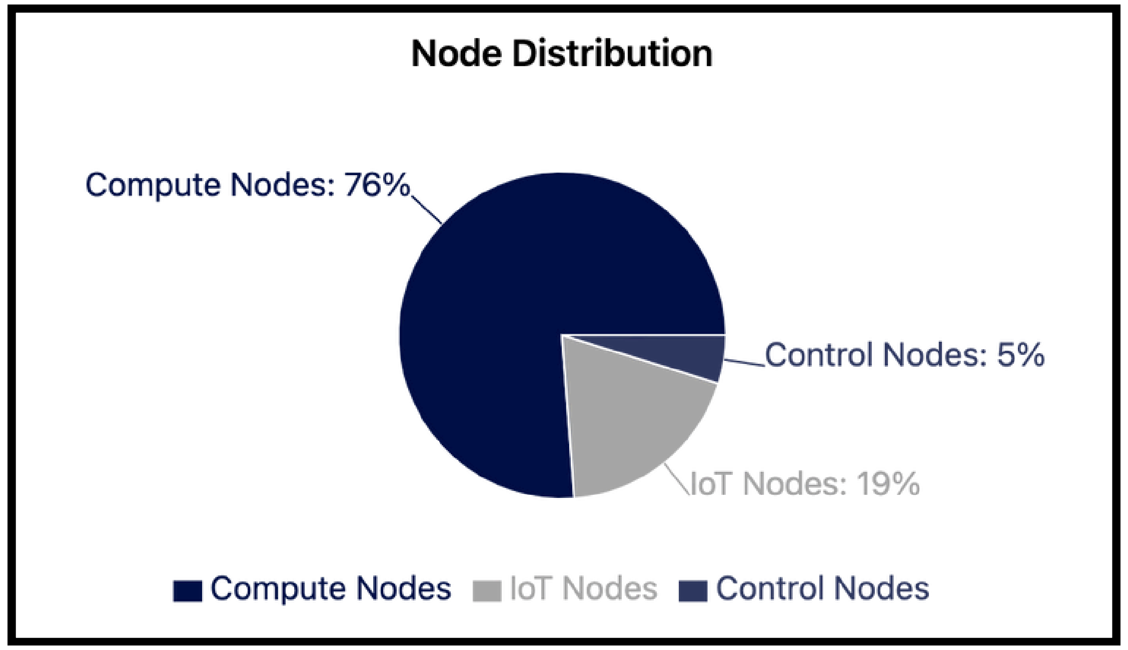
- **Transaction Initiator:** Encapsulates raw data into secure transaction batches
- **Response Manager:** Implements Curve25519 cryptography for secure communication

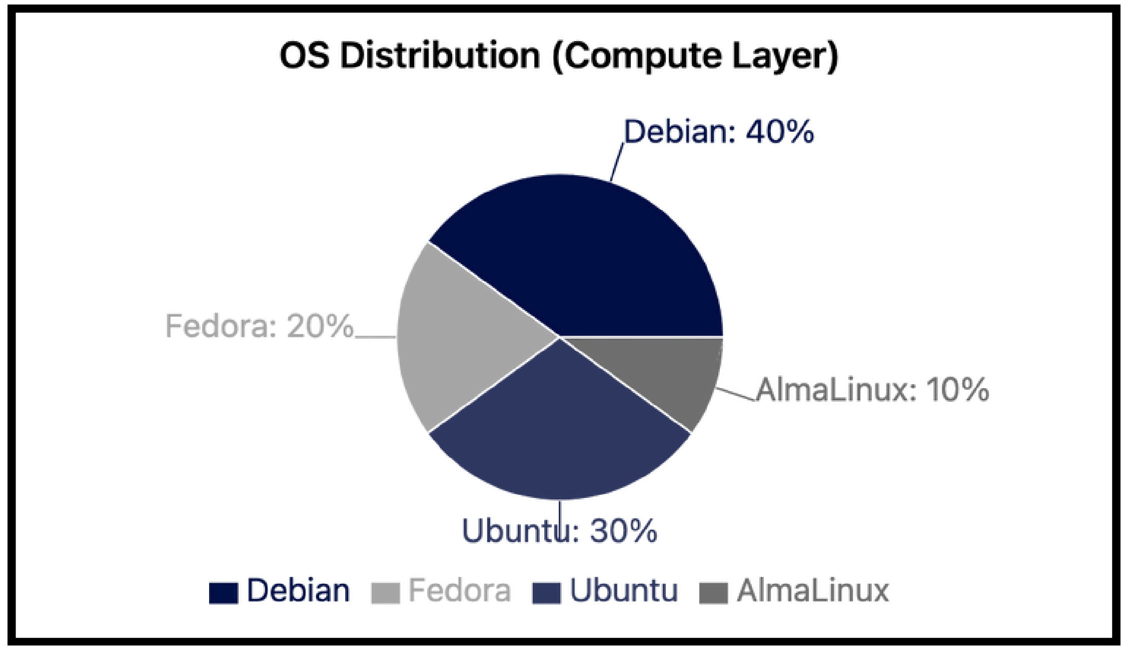
Architectural Benefits

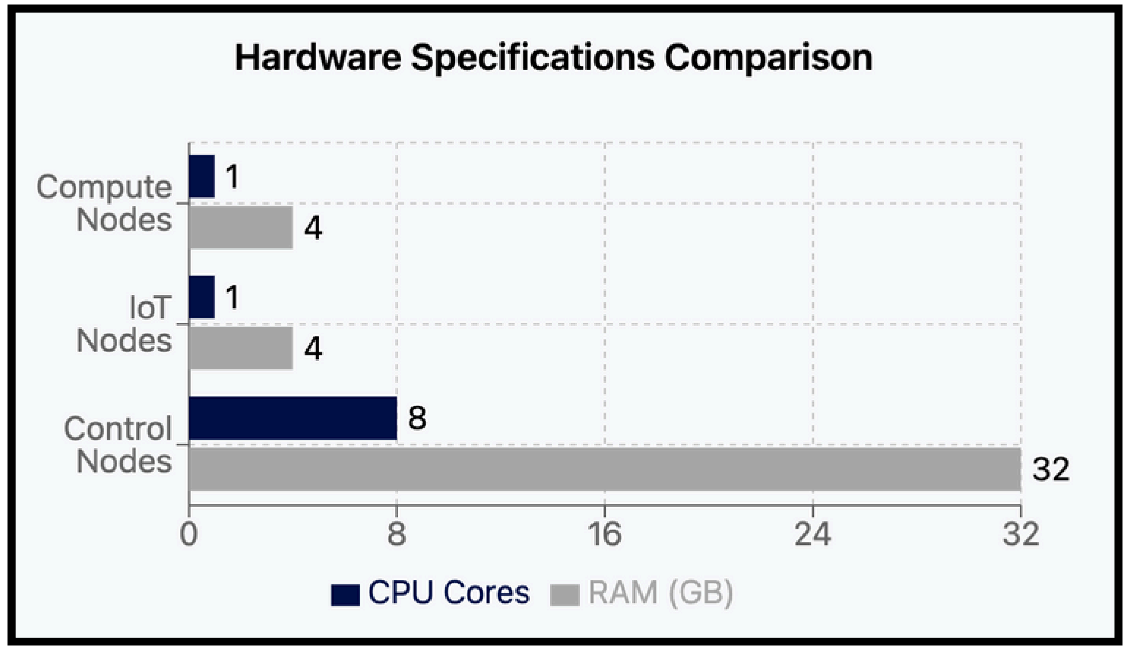
- Lightweight footprint suitable for varied IoT hardware
- Clear separation of data collection and processing concerns
- Adaptable to diverse IoT application requirements

Experimental Setup

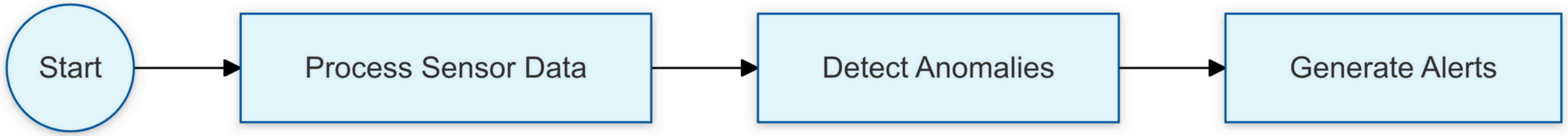
Our Experiments were conducted with a total of **21 nodes** acquired from the **Melbourne Research Cloud**





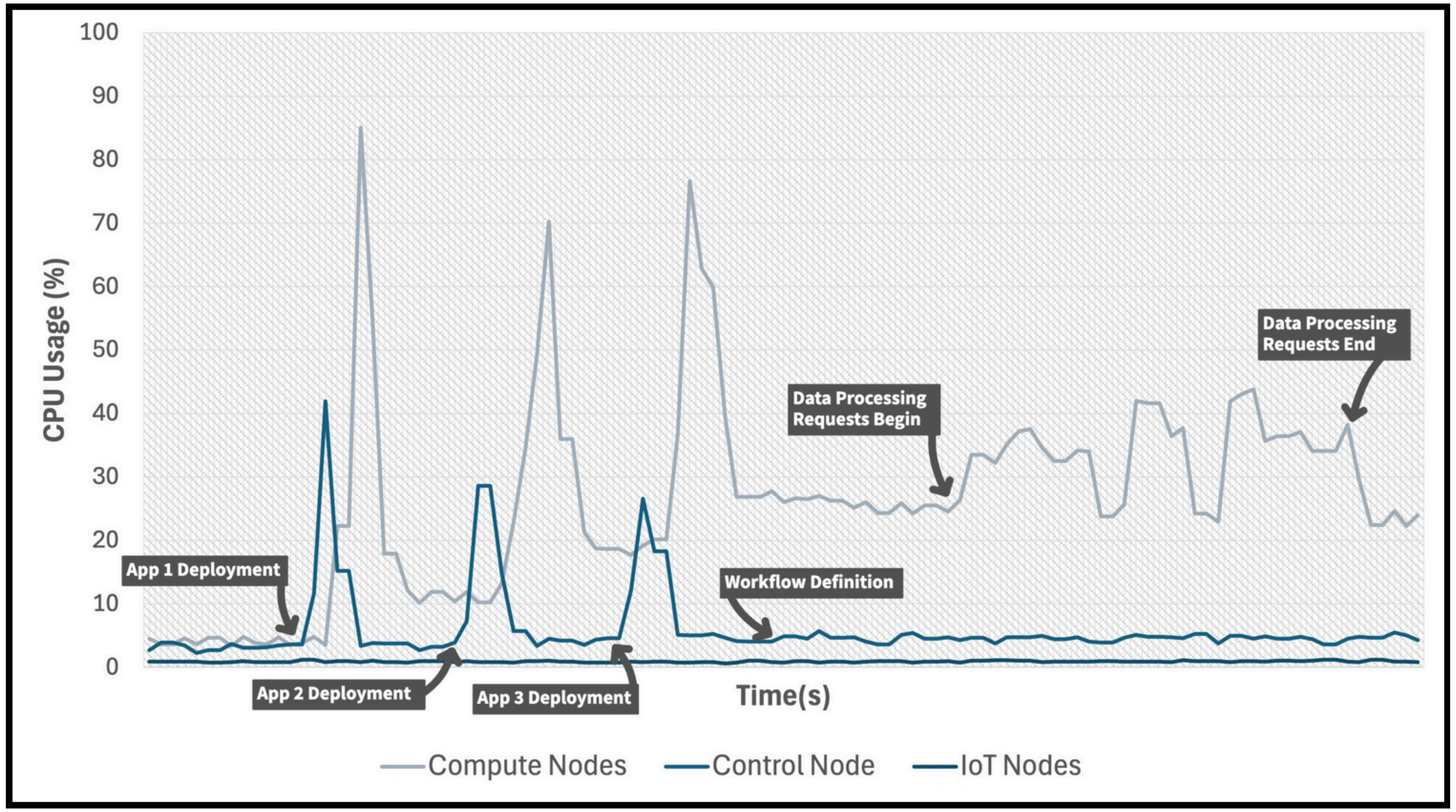


And using a cold-chain monitoring workflow with **3 deployed applications**

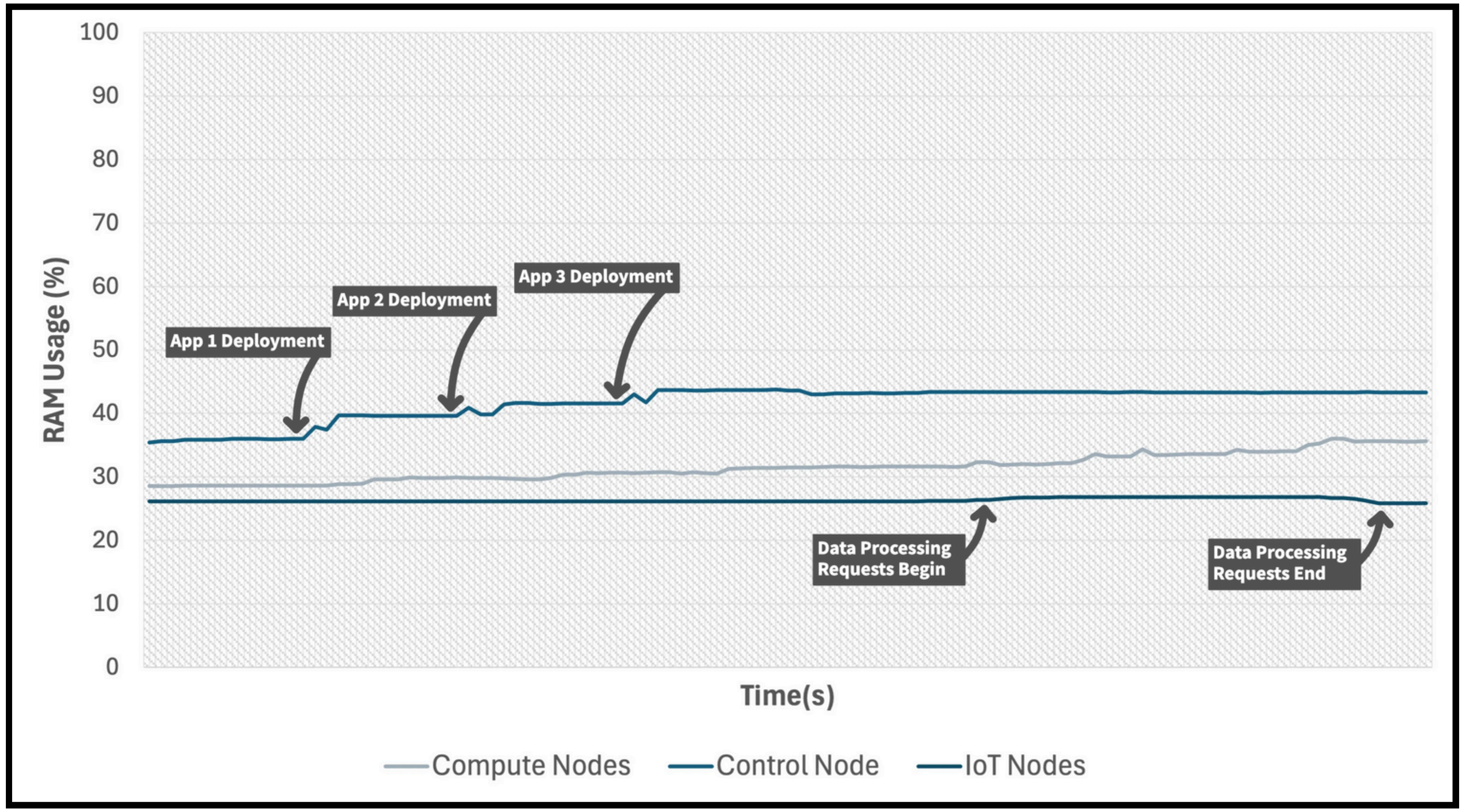


Results

CPU Usage by Workflow Progression



RAM Usage by Workflow Progression

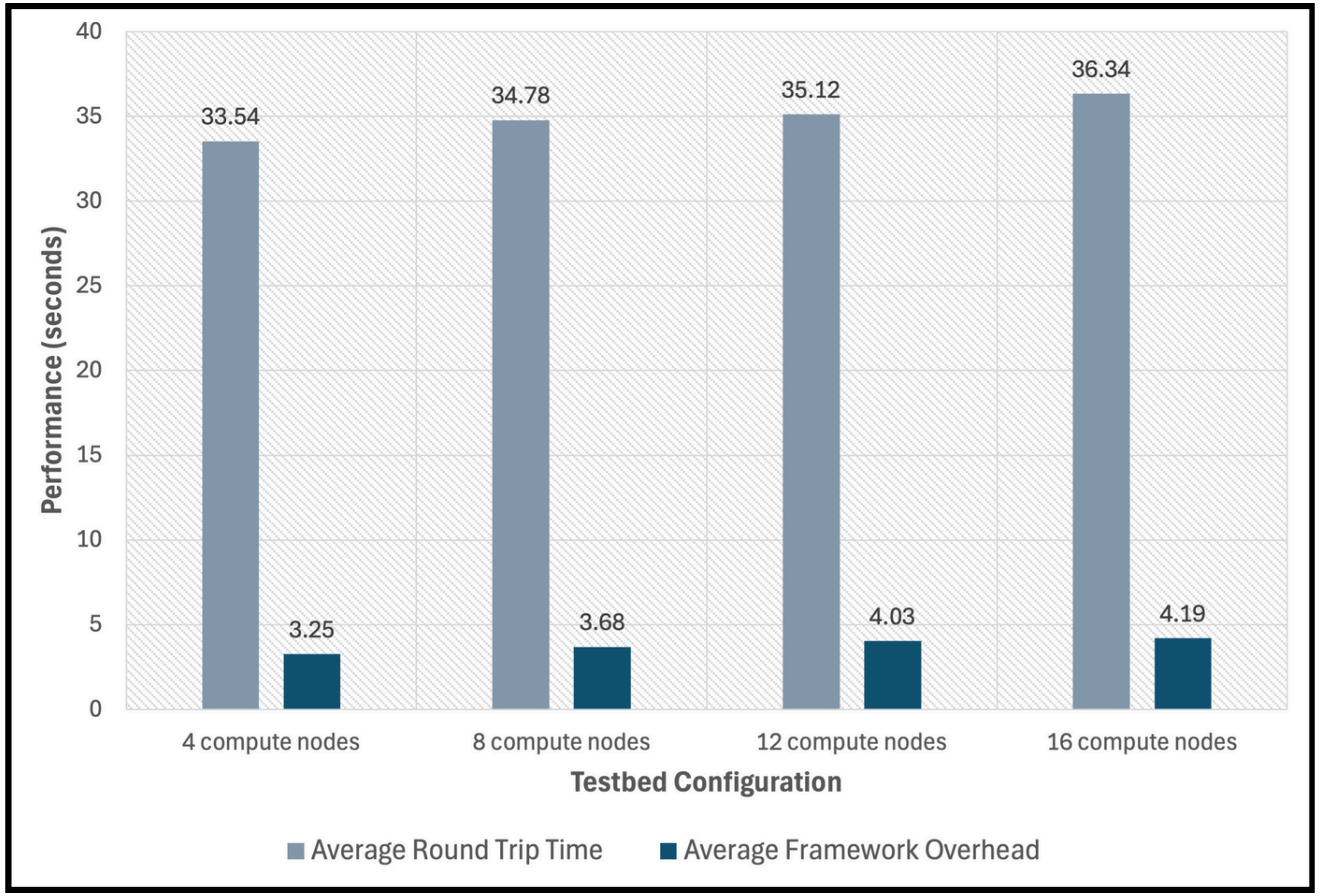


- Staggered deployment peaks show two-phase application distribution mechanism
- Minimal resource utilization in Perception Layer throughout operations
- Stable RAM consumption in Computation Layer during operational phases
- Sustained 45% CPU utilization during concurrent data processing
- Resource patterns demonstrate effective workload distribution across heterogeneous nodes
- Stable baseline consumption shows framework efficiency outside peak operations

Results

THE UNIVERSITY OF MELBOURNE

Performance Comparison by Testbed Configuration



Average Round Trip Time: End-to-end processing duration from initial IoT request to result delivery
Framework Overhead Time: Processing time added by TrustMesh, excluding actual application execution

Framework overhead increases linearly from 3.25s to 4.19s as nodes scale from 4 to 16

Only 8.3% increase in round-trip time despite quadrupling nodes (4→16)

Linear overhead increase observed, though PBFT's $O(n^2)$ complexity suggests this pattern may change with larger node counts

Results

THE UNIVERSITY OF MELBOURNE

Both experiments were conducted with **31.25%** of the network configured to accept malicious scheduling attempts

Metric	Scenario 1	Scenario 2
Detection Latency (ms)	127 ± 15	142 ± 18
Recovery Time (s)	5.23 ± 0.12	5.31 ± 0.15
CPU Utilization (%)	45.5	47.8

Scenario 1: Non-designated nodes attempting to propose schedules (safety property violation)

Scenario 2: Nodes designated to a new request attempting to interfere with already scheduled requests (agreement property violation)

ODENSE • DENMARK
ICSA 2025
31 MARCH - 4 APRIL
Organized by SDU Software Engineering SDU

MELBOURNE CLOUDS LAB

THE UNIVERSITY OF MELBOURNE

Q&A

Paper Preprint

LinkedIn

THE UNIVERSITY OF MELBOURNE

Presentation Link