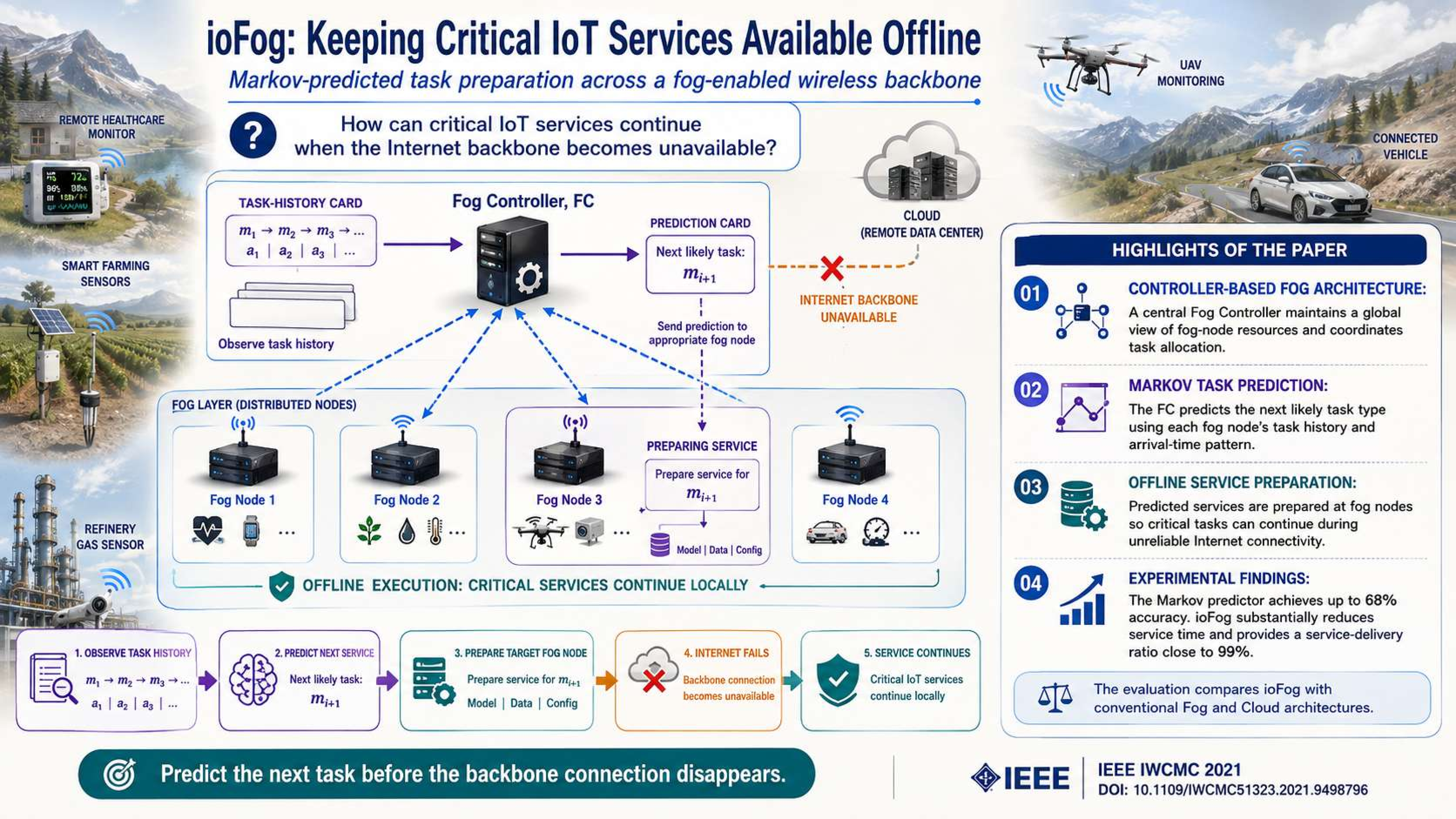
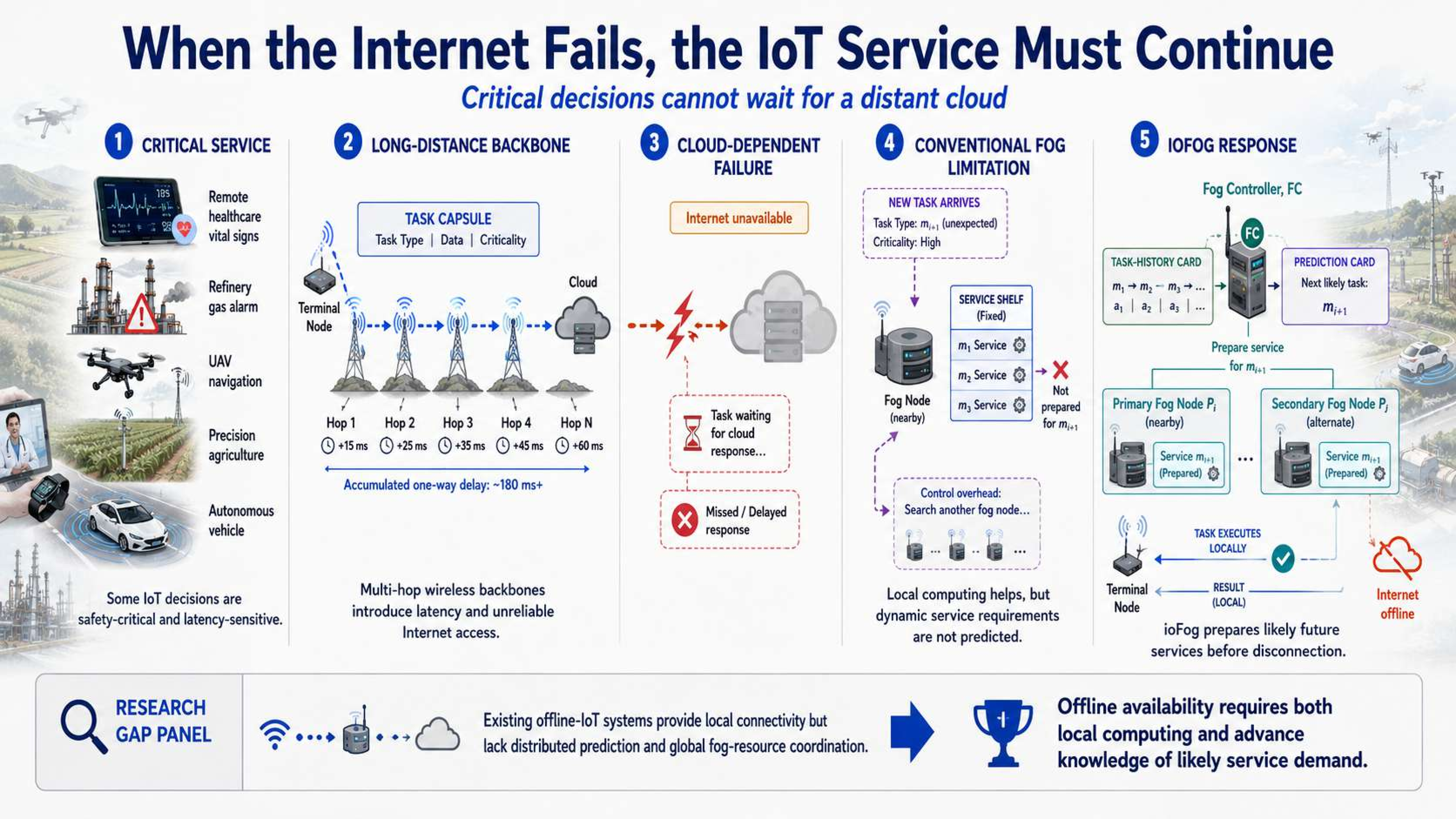
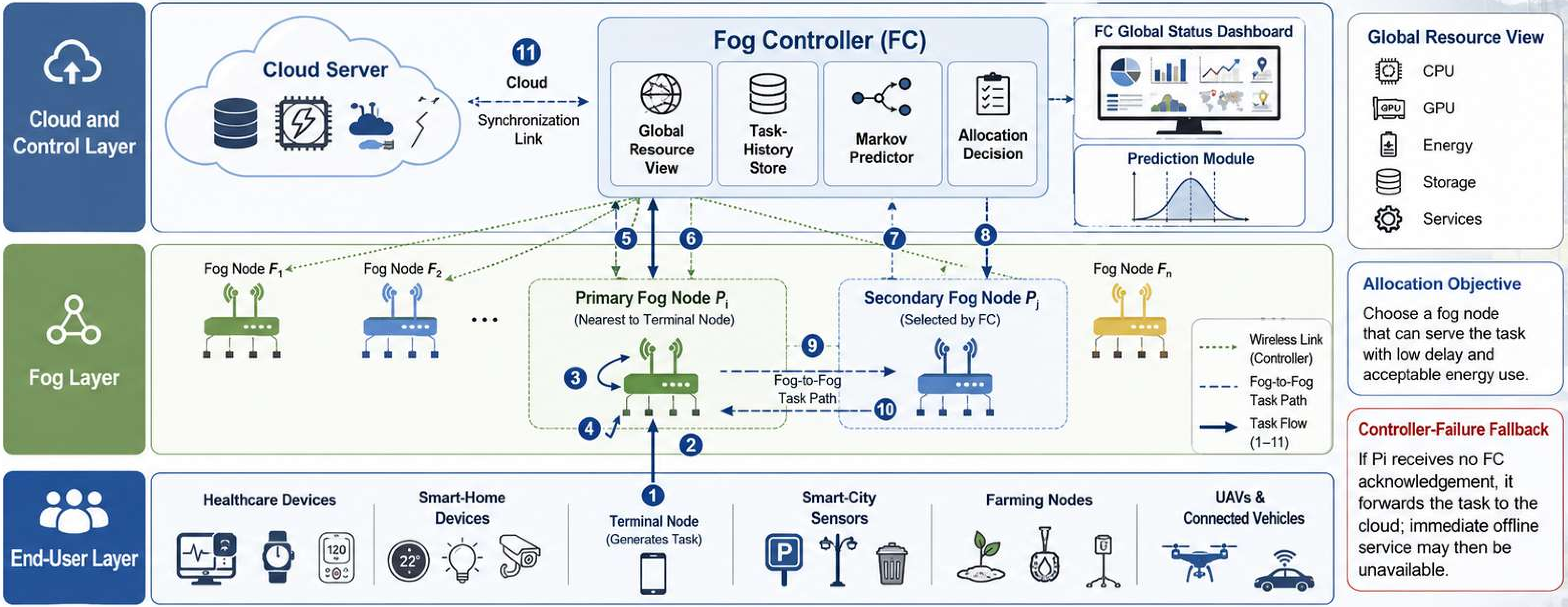


ioFog: Keeping Critical IoT Services Available Offline
Markov-predicted task preparation across a fog-enabled wireless backbone
How can critical IoT services continue when the Internet backbone becomes unavailable?
REMOTE HEALTHCARE MONITOR
SMART FARMING SENSORS
REFINERY GAS SENSOR
UAV MONITORING
CONNECTED VEHICLE
TASK-HISTORY CARD
$m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow \ldots$
a_1 | a_2 | a_3 | ...
Observe task history
Fog Controller, FC
PREDICTION CARD
Next likely task: m_{i+1}
Send prediction to appropriate fog node
CLOUD (REMOTE DATA CENTER)
INTERNET BACKBONE UNAVAILABLE
FOG LAYER (DISTRIBUTED NODES)
Fog Node 1
Fog Node 2
Fog Node 3
PREPARING SERVICE
Prepare service for m_{i+1}
Model | Data | Config
Fog Node 4
OFFLINE EXECUTION: CRITICAL SERVICES CONTINUE LOCALLY
1. OBSERVE TASK HISTORY
$m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow \ldots$
a_1 | a_2 | a_3 | ...
2. PREDICT NEXT SERVICE
Next likely task: m_{i+1}
3. PREPARE TARGET FOG NODE
Prepare service for m_{i+1}
Model | Data | Config
4. INTERNET FAILS
Backbone connection becomes unavailable
5. SERVICE CONTINUES
Critical IoT services continue locally
Predict the next task before the backbone connection disappears.
HIGHLIGHTS OF THE PAPER
01 CONTROLLER-BASED FOG ARCHITECTURE:
A central Fog Controller maintains a global view of fog-node resources and coordinates task allocation.
02 MARKOV TASK PREDICTION:
The FC predicts the next likely task type using each fog node's task history and arrival-time pattern.
03 OFFLINE SERVICE PREPARATION:
Predicted services are prepared at fog nodes so critical tasks can continue during unreliable Internet connectivity.
04 EXPERIMENTAL FINDINGS:
The Markov predictor achieves up to 68% accuracy. ioFog substantially reduces service time and provides a service-delivery ratio close to 99%.
The evaluation compares ioFog with conventional Fog and Cloud architectures.
IEEE
IEEE IWCMC 2021
DOI: 10.1109/IWCMC51323.2021.9498796


When the Internet Fails, the IoT Service Must Continue
Critical decisions cannot wait for a distant cloud
1 CRITICAL SERVICE
Remote healthcare vital signs
Refinery gas alarm
UAV navigation
Precision agriculture
Autonomous vehicle
Some IoT decisions are safety-critical and latency-sensitive.
2 LONG-DISTANCE BACKBONE
TASK CAPSULE
Task Type | Data | Criticality
Terminal Node
Cloud
Hop 1 +15 ms
Hop 2 +25 ms
Hop 3 +35 ms
Hop 4 +45 ms
Hop N +60 ms
Accumulated one-way delay: ~180 ms+
Multi-hop wireless backbones introduce latency and unreliable Internet access.
3 CLOUD-DEPENDENT FAILURE
Internet unavailable
Task waiting for cloud response...
Missed / Delayed response
4 CONVENTIONAL FOG LIMITATION
NEW TASK ARRIVES
Task Type: m_{i+1} (unexpected)
Criticality: High
SERVICE SHELF (Fixed)
m_1 Service
m_2 Service
m_3 Service
Not prepared for m_{i+1}
Fog Node (nearby)
Control overhead: Search another fog node...
Local computing helps, but dynamic service requirements are not predicted.
5 IOFOG RESPONSE
Fog Controller, FC
FC
TASK-HISTORY CARD
$m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow \dots$
a_1 | a_2 | a_3 | ...
PREDICTION CARD
Next likely task:
m_{i+1}
Prepare service for m_{i+1}
Primary Fog Node P_i (nearby)
Service m_{i+1} (Prepared)
Secondary Fog Node P_j (alternate)
Service m_{i+1} (Prepared)
TASK EXECUTES LOCALLY
Terminal Node
RESULT (LOCAL)
Internet offline
ioFog prepares likely future services before disconnection.
RESEARCH GAP PANEL
Existing offline-IoT systems provide local connectivity but lack distributed prediction and global fog-resource coordination.
Offline availability requires both local computing and advance knowledge of likely service demand.

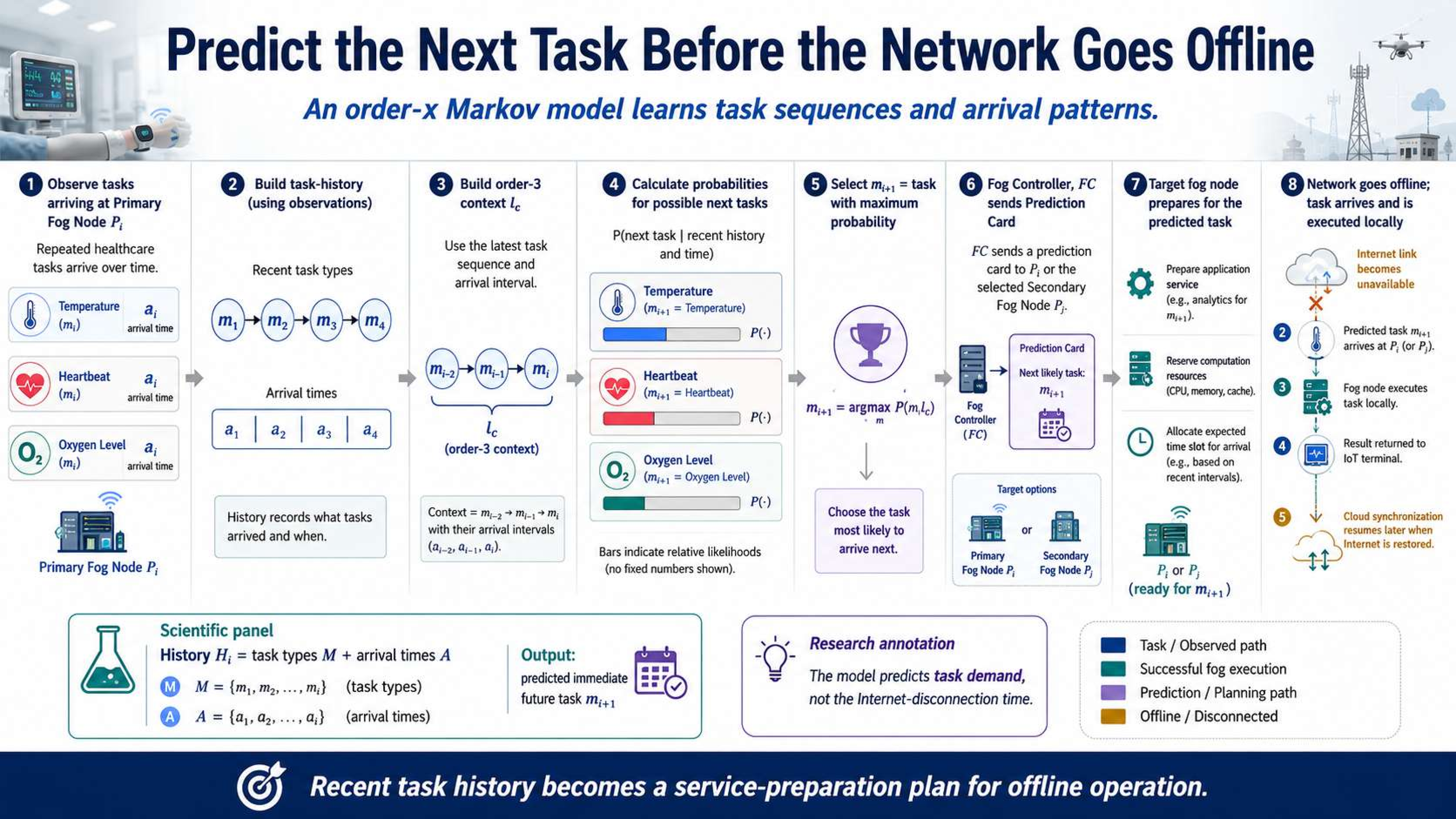
Inside the ioFog Architecture

Global fog awareness with local offline execution



1. Terminal node generates a task.
2. Task reaches nearest Primary Fog Node P_i.
3. P_i checks whether it can execute the task.
4. If P_i is suitable, execute locally.
5. P_i sends updated resource status to FC.
6. If P_i cannot execute, request allocation from FC.
7. FC checks global fog-resource view.
8. FC selects Secondary Fog Node P_j.
9. P_i forwards the task to P_j.
10. P_j computes and returns the result.
11. Cloud receives a copy or synchronization when connectivity is available.

FC coordinates globally, while fog nodes execute locally.



Predict the Next Task Before the Network Goes Offline
An order-x Markov model learns task sequences and arrival patterns.
1 Observe tasks arriving at Primary Fog Node P_i
Repeated healthcare tasks arrive over time.
Temperature (m_i) a_i arrival time
Heartbeat (m_i) a_i arrival time
Oxygen Level (m_i) a_i arrival time
O_2
Primary Fog Node P_i
2 Build task-history (using observations)
Recent task types
$m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow m_4$
Arrival times
a_1 | a_2 | a_3 | a_4
History records what tasks arrived and when.
3 Build order-3 context l_c
Use the latest task sequence and arrival interval.
$\underbrace{m_{i-2} \rightarrow m_{i-1} \rightarrow m_i}_{l_c}$
(order-3 context)
Context = $m_{i-2} \rightarrow m_{i-1} \rightarrow m_i$ with their arrival intervals (a_{i-2}, a_{i-1}, a_i).
4 Calculate probabilities for possible next tasks
P(next task | recent history and time)
Temperature (m_{i+1} = Temperature) $P(\cdot)$
Heartbeat (m_{i+1} = Heartbeat) $P(\cdot)$
Oxygen Level (m_{i+1} = Oxygen Level) $P(\cdot)$
Bars indicate relative likelihoods (no fixed numbers shown).
5 Select m_{i+1} = task with maximum probability
$m_{i+1} = \underset{m}{\operatorname{argmax}}\, P(m, l_c)$
Choose the task most likely to arrive next.
6 Fog Controller, FC sends Prediction Card
FC sends a prediction card to P_i or the selected Secondary Fog Node P_j.
Fog Controller (FC)
Prediction Card
Next likely task: m_{i+1}
Target options
Primary Fog Node P_i
or
Secondary Fog Node P_j
7 Target fog node prepares for the predicted task
Prepare application service (e.g., analytics for m_{i+1}).
Reserve computation resources (CPU, memory, cache).
Allocate expected time slot for arrival (e.g., based on recent intervals).
P_i or P_j (ready for m_{i+1})
8 Network goes offline; task arrives and is executed locally
Internet link becomes unavailable
2 Predicted task m_{i+1} arrives at P_i (or P_j).
3 Fog node executes task locally.
4 Result returned to IoT terminal.
5 Cloud synchronization resumes later when Internet is restored.
Scientific panel
History H_i = task types M + arrival times A
M $M = \{m_1, m_2, \dots, m_i\}$ (task types)
A $A = \{a_1, a_2, \dots, a_i\}$ (arrival times)
Output:
predicted immediate future task m_{i+1}
Research annotation
The model predicts task demand, not the Internet-disconnection time.
Task / Observed path
Successful fog execution
Prediction / Planning path
Offline / Disconnected
Recent task history becomes a service-preparation plan for offline operation.

What the ioFog Experiments Show

Prediction accuracy, service time and offline reliability

1 Task-Prediction Accuracy

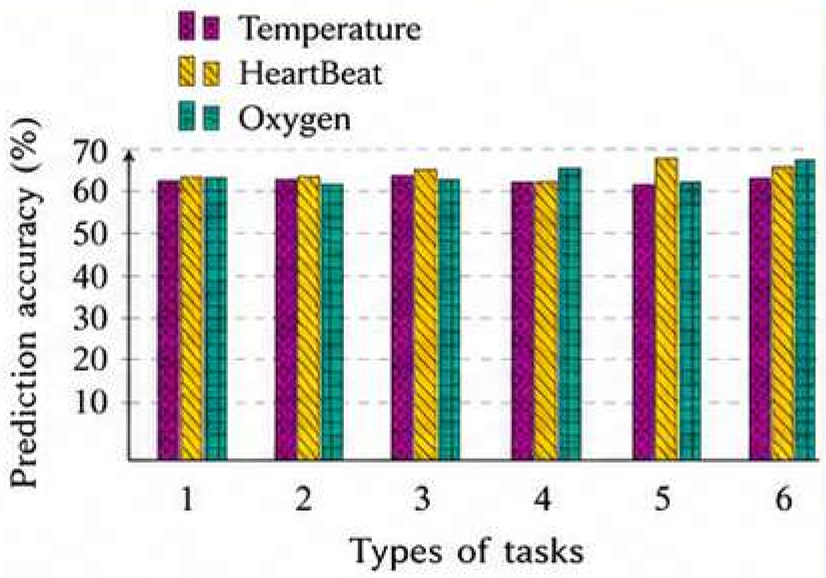


Fig. 4. Prediction Accuracy

✔ Up to 68% prediction accuracy

2 Average Service Time

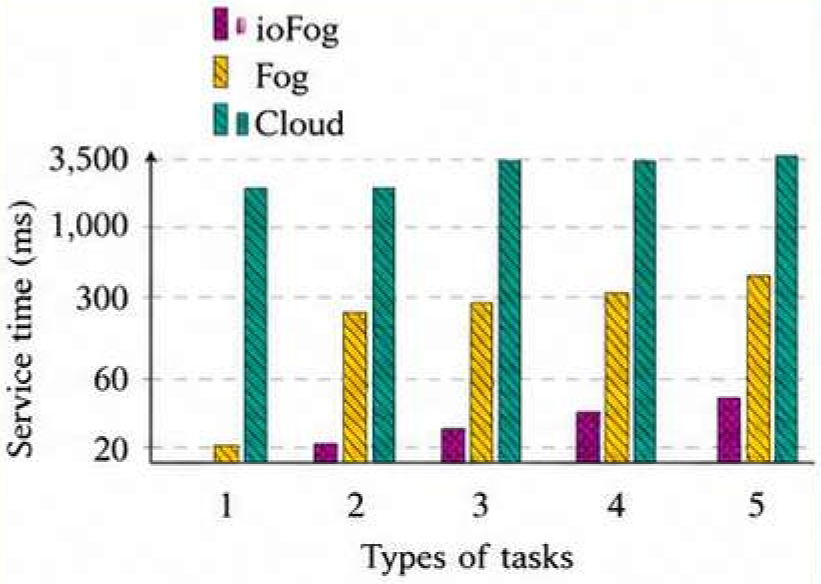


Fig. 5. Average Service Time

✔ ioFog provides the lowest service time

3 Offline Service Delivery

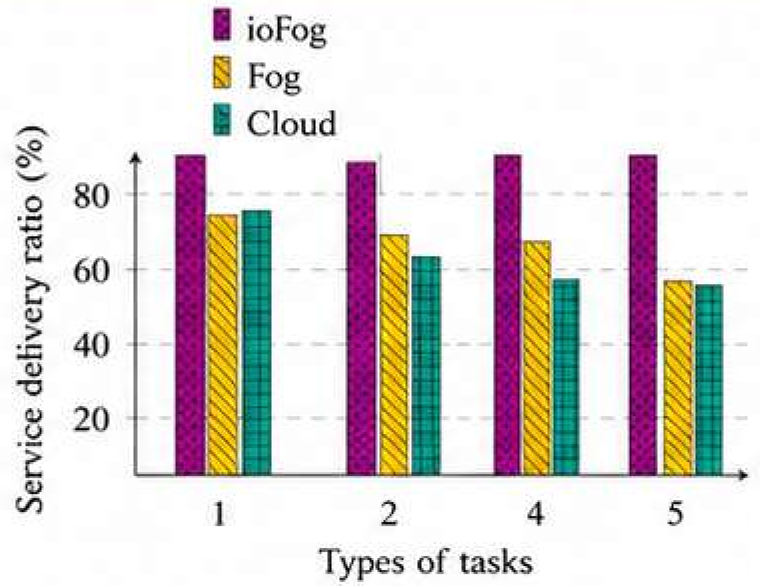


Fig. 6. Average Service Delivery Ratio

✔ Almost 99% service-delivery ratio

Simulation Setup

- iFogSim
- dynamic service requests
- Wi-Fi-based long-distance point-to-point backbone
- multiple backbone levels
- three healthcare task types
- Temperature
- Heartbeet
- Oxygen Level
- Comparison: ioFog, Fog and Cloud

Conclusion Chain

Global fog status → task-history learning → Markov prediction → advance service preparation → offline fog execution → lower service time and higher availability

Current Limitations

The centralized FC introduces a single-point-of-failure concern. Prediction accuracy is moderate, and failure to receive an FC acknowledgement may force cloud fallback.

Future Work

- improve task-prediction accuracy
- use machine-learning methods for prediction and resource optimization
- deploy ioFog on a real IoT testbed

Research Conclusion

ioFog combines global fog coordination with Markov-based task prediction to maintain critical IoT services during unreliable Internet connectivity.

Observe the demand. Predict the service. Keep the IoT running offline.