Failure Prediction in the Internet of Things due to Memory Exhaustion

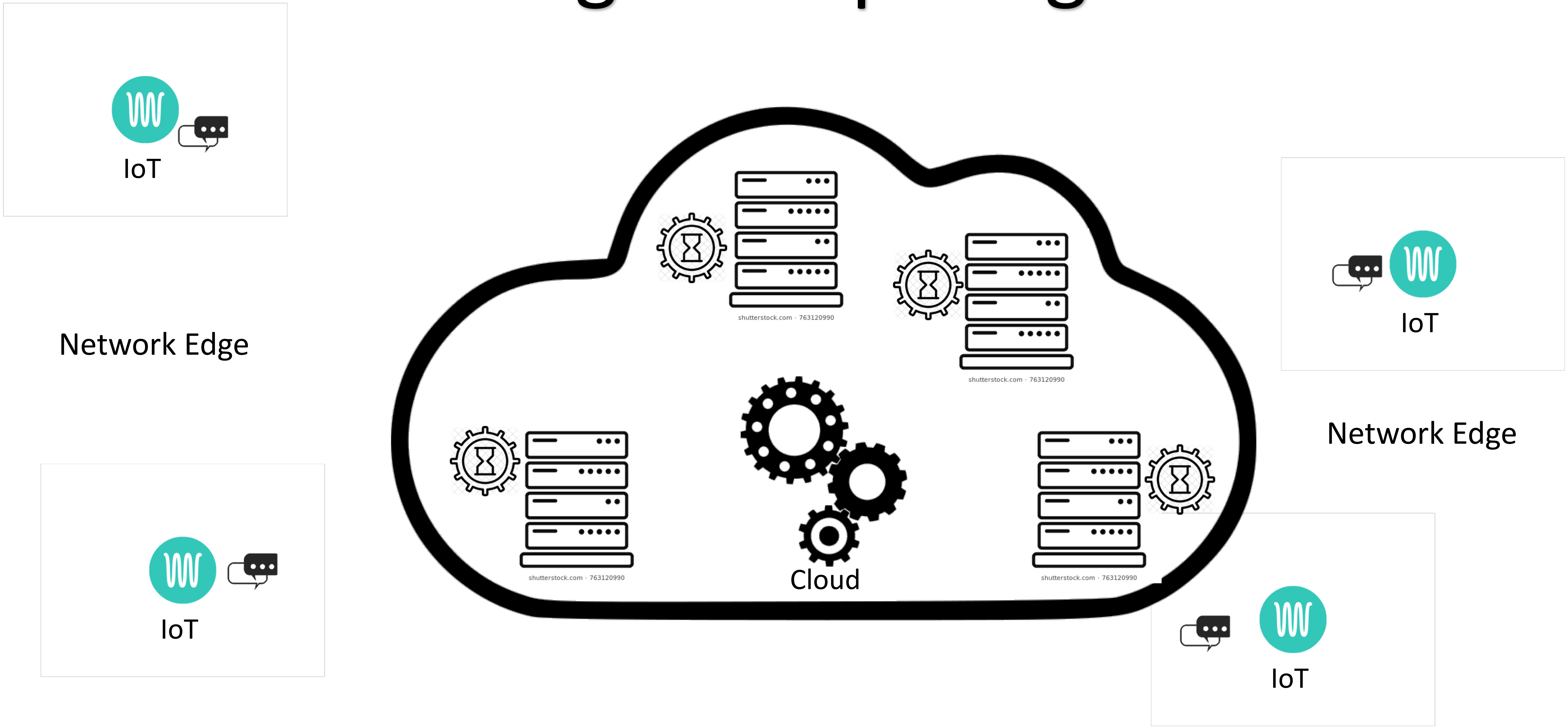
Mohammad Rafiuzzaman[1], Julien Gascon-Samson[2],

Karthik Pattabiraman[1], Sathish Gopalakrishnan[1]

[1]Electrical and Computer Engineering, The University of British Columbia (UBC)

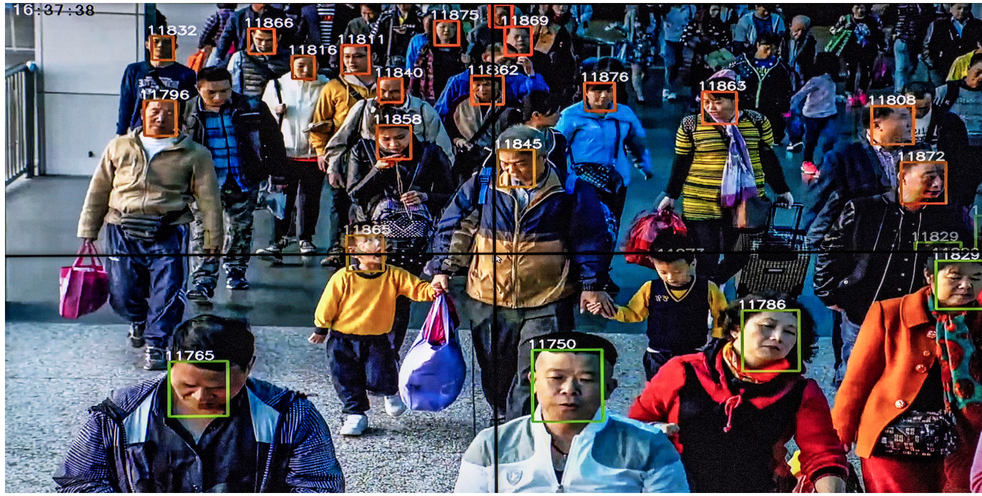
[2]École de technologie supérieure (ÉTS), Montreal

Edge Computing



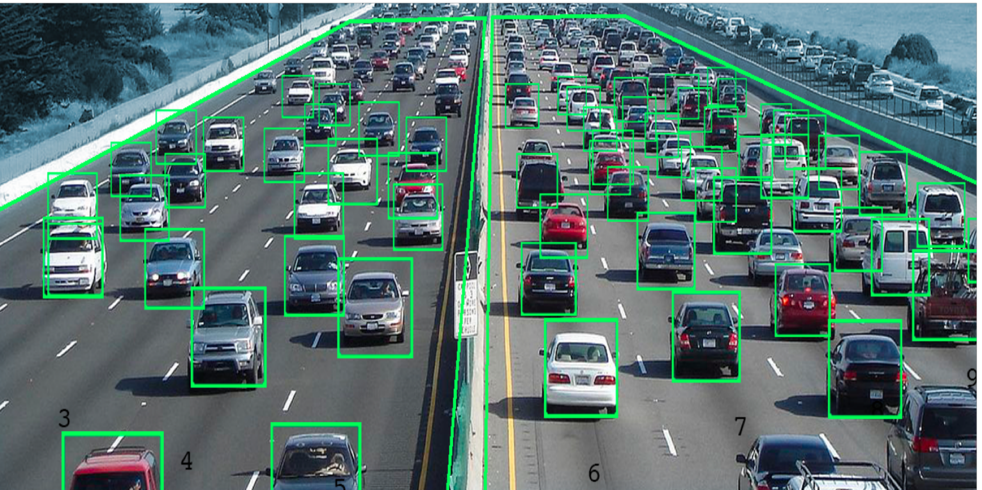
An Example Application

www.zdnet.com



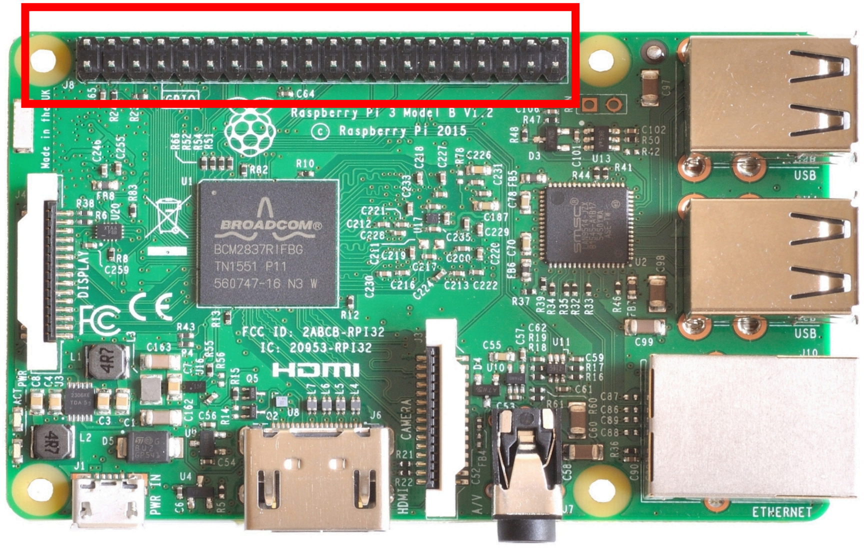
The New York Times

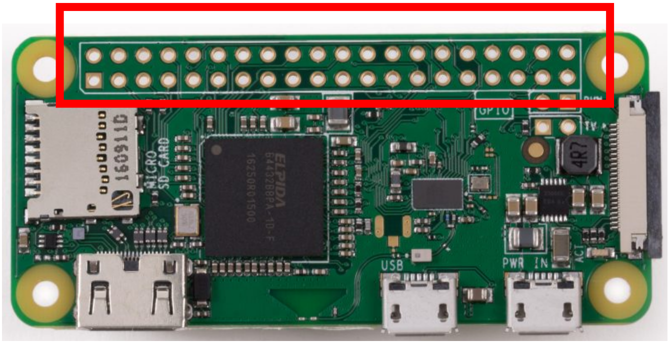
www.trafficvision.com



The Washington Post

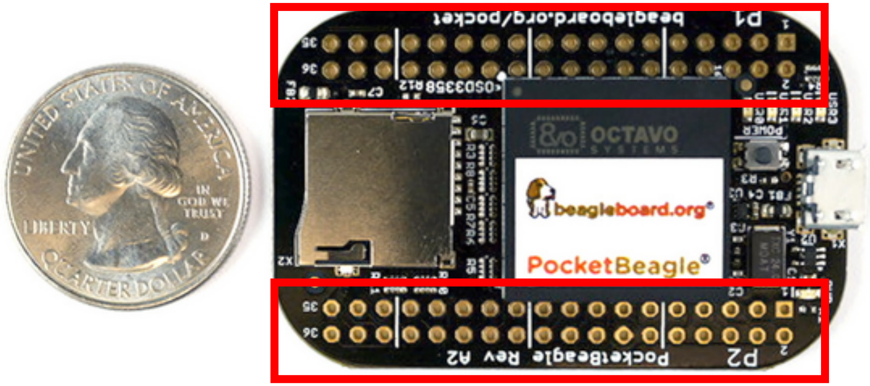
Today’s IoT





Sensor/Actuator

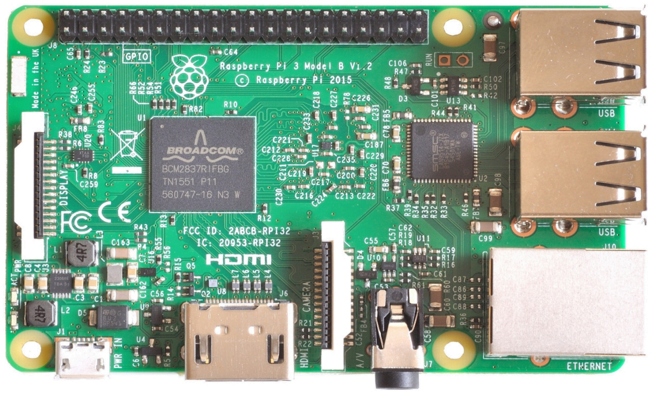
High-level
Applications



GPOS

Challenges

- **Resource (e.g., memory) constraints**
  - Resource constraints
  - Performance uncertainty
  - Unexpected failures
- General purpose applications
- Device specific optimizations
- Expected availability



Contributions

✓ **Define a systematic approach to identify memory-failures in IoT**

- Develop a novel technique called MARK for handling such failures
- Introduce simple classification (k-NN) models to predict such failures
- Evaluate those models under various real-world circumstances

Primary Resource Bottleneck

Resource consumption can be:

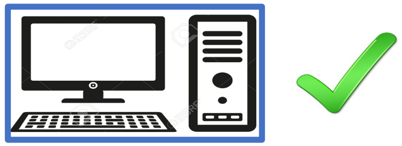
1. CPU-bound (processes waiting on CPU resources),

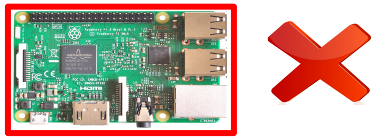
Memory-bound (processes consuming memory),

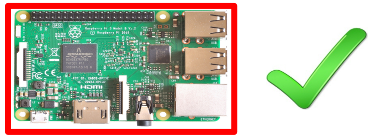
3. I/O bound (processes competing for disk or network I/O).

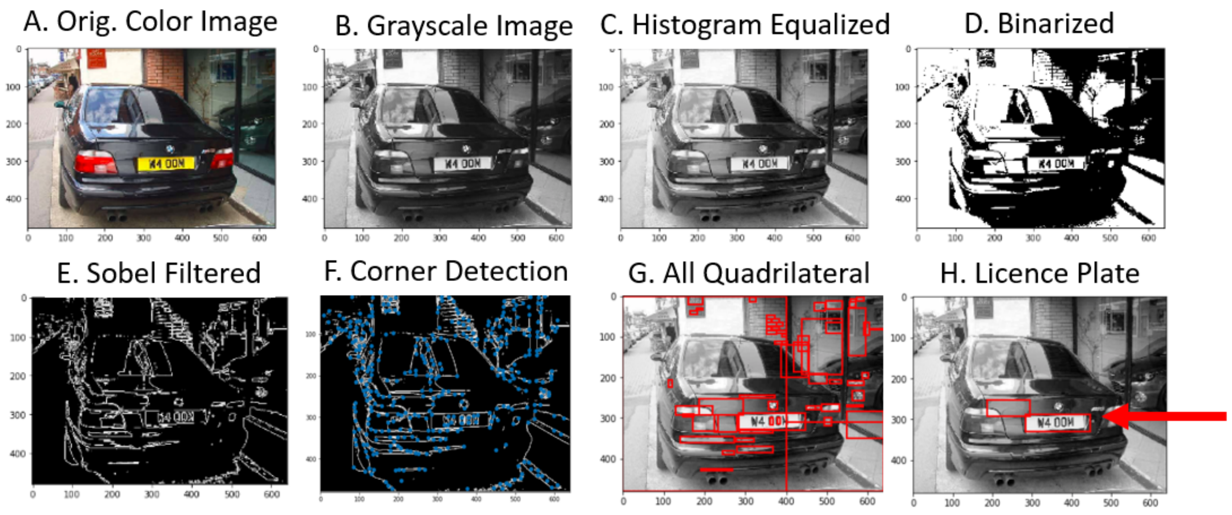
Many of the devices are severely memory constrained (< 1 GB typically)

Failures In Edge Devices



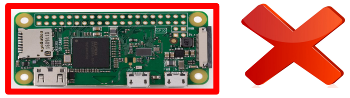


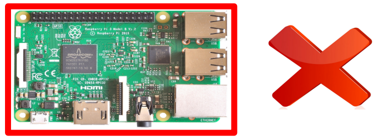


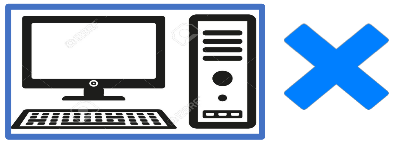


Failures:

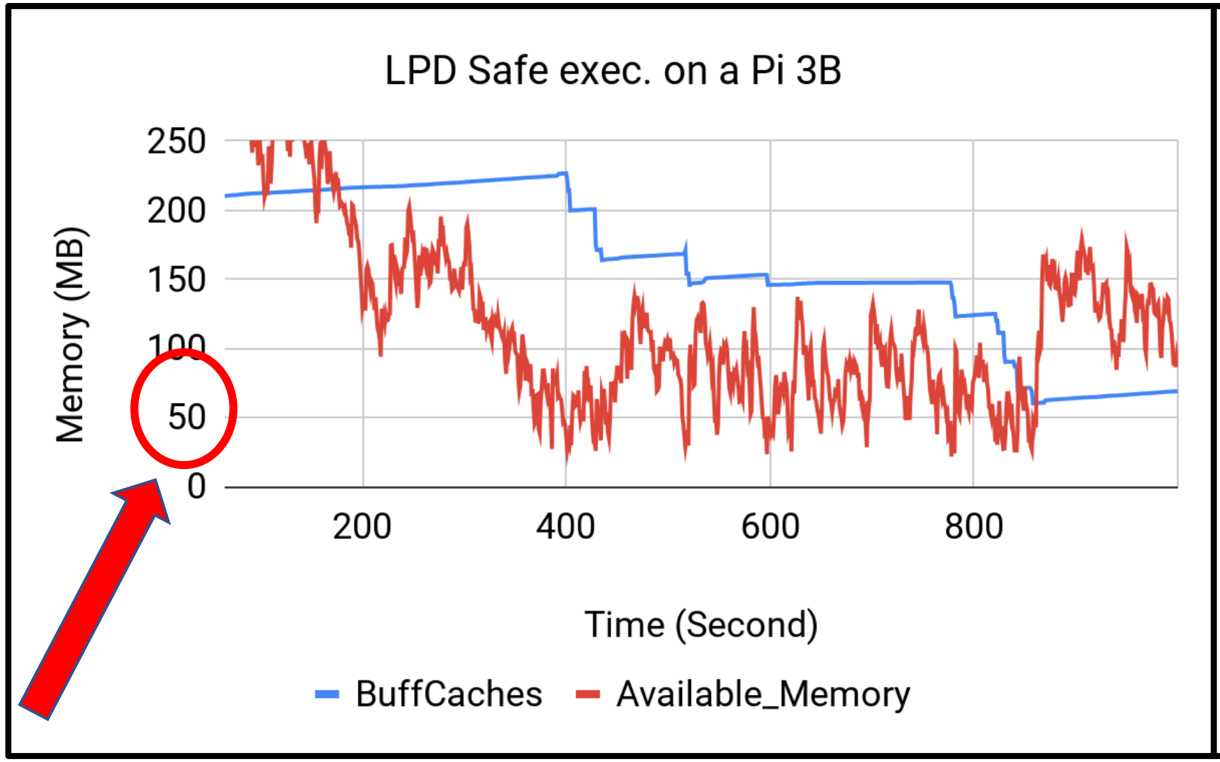
1. Unresponsive application
2. Application crash
3. Permanent unresponsive system

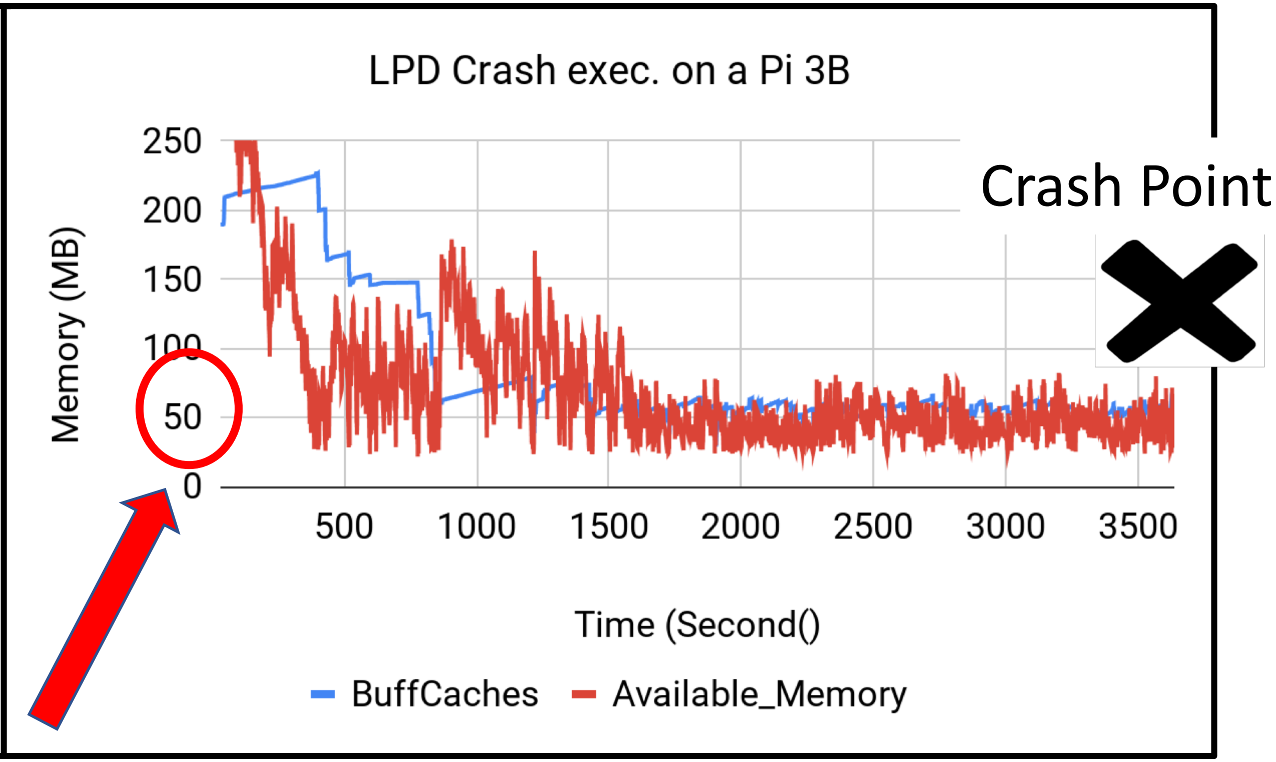




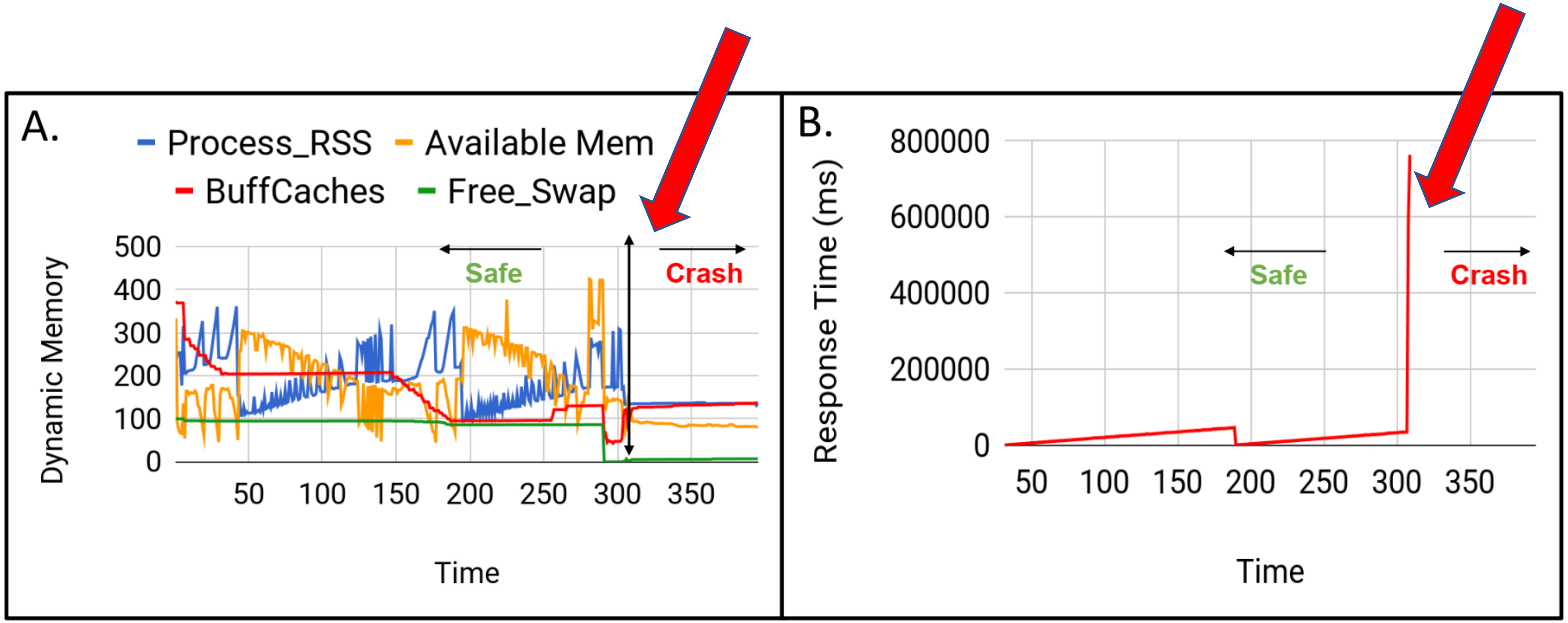


Problem with Threshold-based Approaches





Preliminary Study: Memory Failures

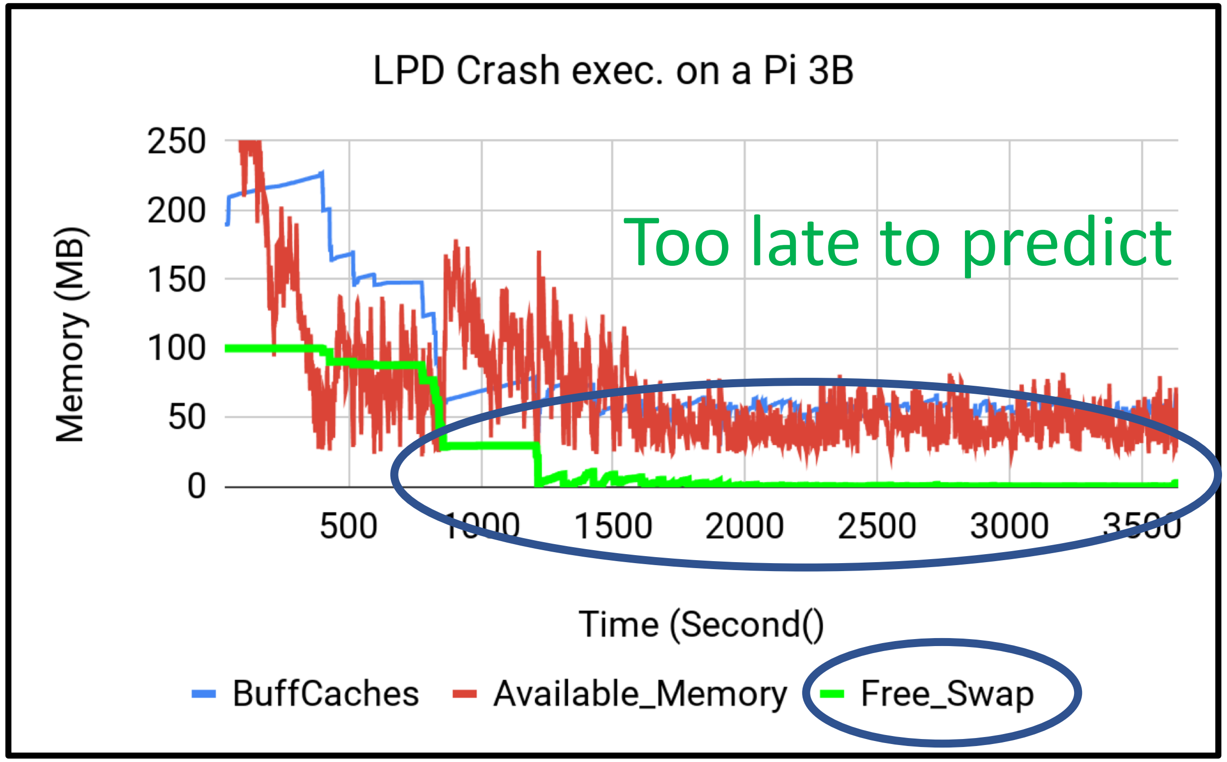


Preliminary Study: Failure Indicators

Spearman's Rank-Order Correlation:

- ✓ Reserved system memory
- ✗ Available swap space
- ✓ System CPU usage

Final memory-failures indicators



Contributions

- Define a systematic approach to identify memory-failures in IoT.
- **Develop a novel technique called MARK for handling such failures.**
- Introduce simple classification (k-NN) models to predict such failures.
- Evaluate those models under various real-world circumstances.

Features of MARK

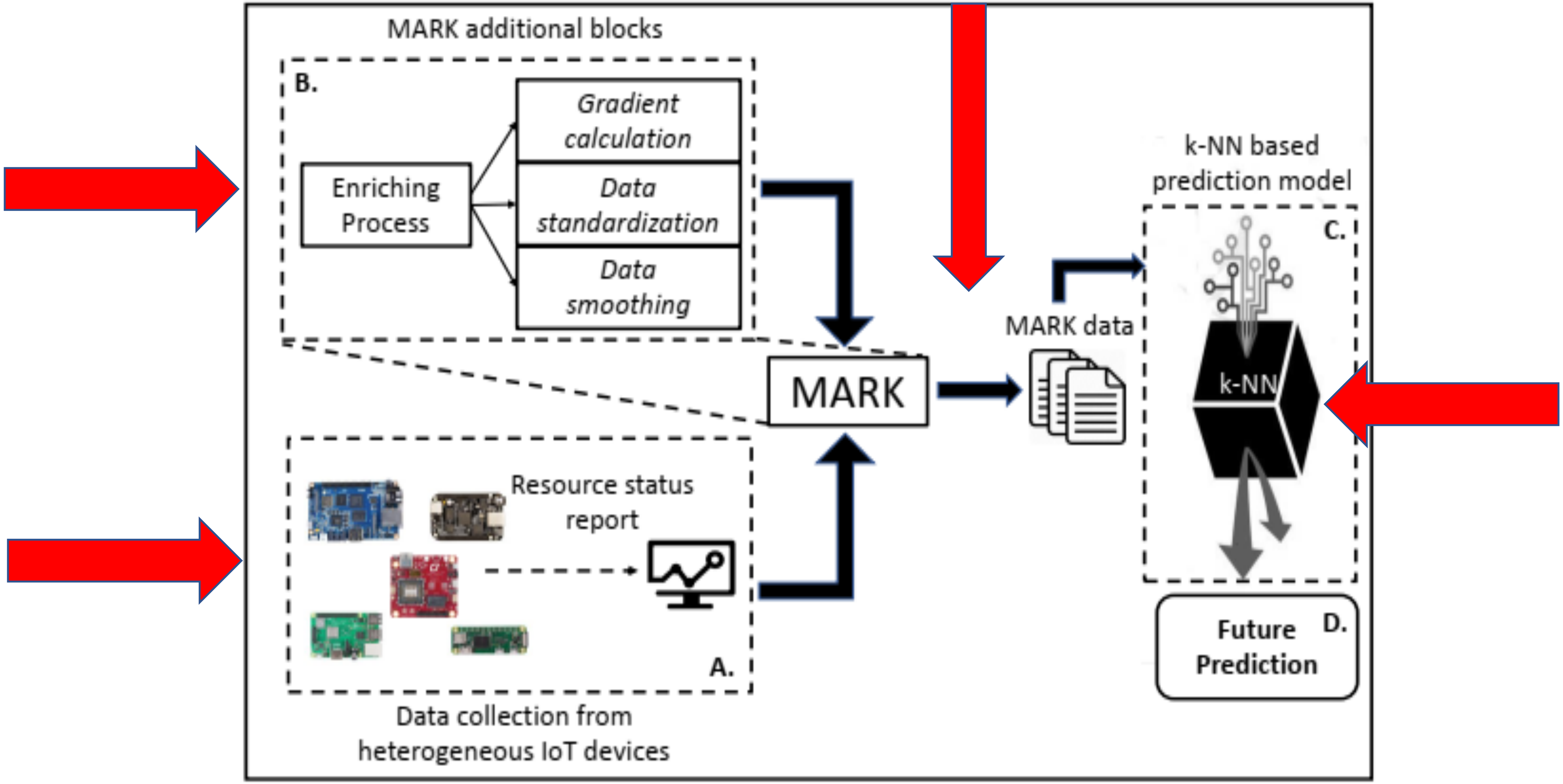
A memory failure prediction technique called MARK:

- Predicts onset of memory failures.
- Observes failure indicative parameters.
- Applies cross-platform predictor.
- Handles high-level applications.

For resource limited modern IoT systems

Provides enough lead time to prevent memory failures in advance

MARK Workflow



Contributions

- Define a systematic approach to identify memory-failures in IoT.
- Develop a novel technique called MARK for handling such failures.
- **Introduce simple classification (k-NN) models to predict such failures.**
- Evaluate those models under various real-world circumstances.

Prediction model of MARK

Goal:

$$f : R \to S$$

Where:

$$R = \{Resource\ Measurements\}$$

$$S = \{safe, fail\}$$

$$P(Y_{t+LAW} = s_{t+LAW}) = \frac{1}{k} \sum_{i \in A_t} I(Y_{i,t+LAW} = s_{t+LAW})$$

Contributions

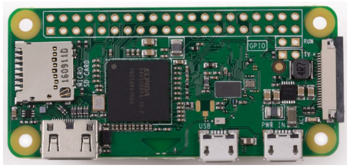
- Define a systematic approach to identify memory-failures in IoT.
- Develop a novel technique called MARK for handling such failures.
- Introduce simple classification (k-NN) models to predict such failures.
- **Evaluate those models under various real-world circumstances.**

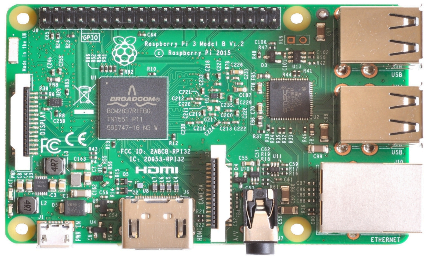
Experimental Setup: Applications and Platforms

Video Surveillance
(node.js) Gascon-Samson et al, 2018.

Automatic LPD
(Python-skimage) Stefan et al. 2014

Sensor Data Processing
(node.js) A real world sensor-server simulation



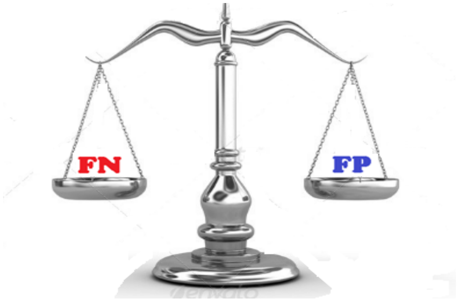


	Memory	*Swap Space*	*Processor*
Pi 0W	512MB	100MB	single-core 1 Ghz ARM6
Pi 3B	1GB	100MB	quad-core 1.2 Ghz ARM7
EC2 t2	1GB	N/A	single-core virtual CPU

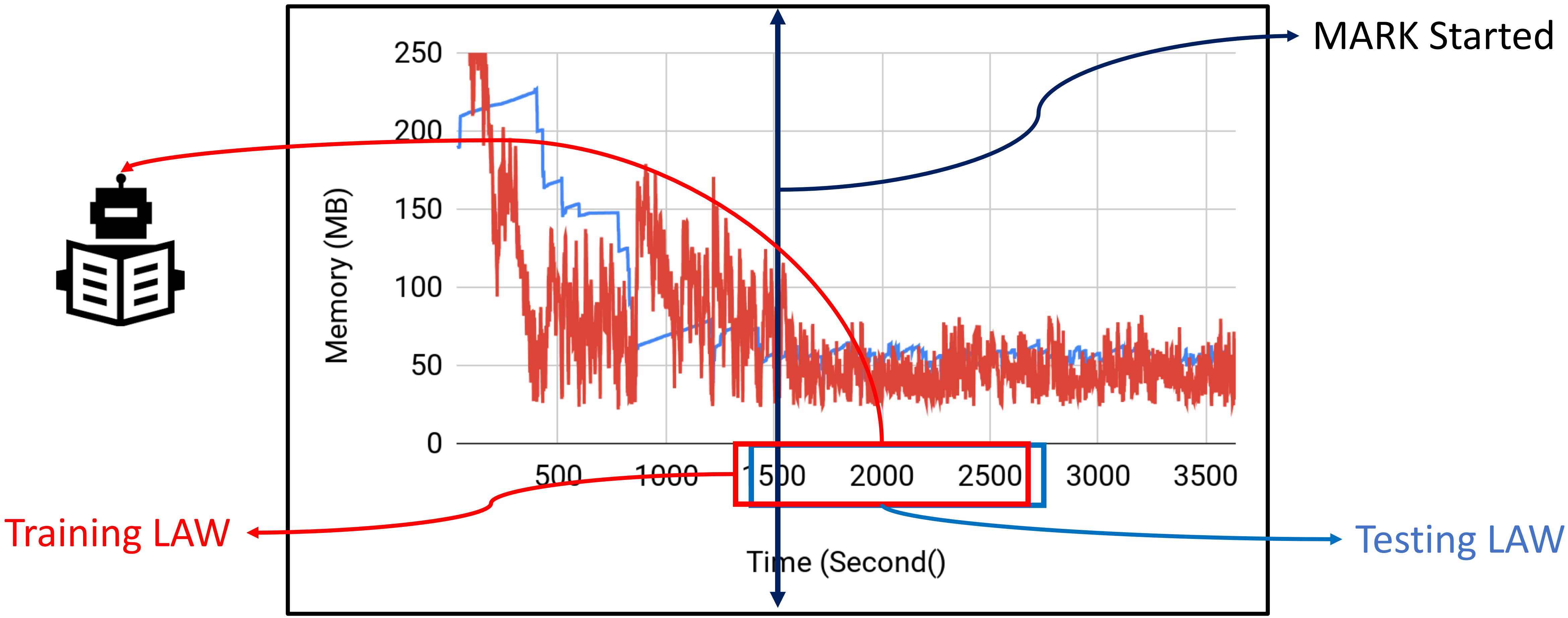
Experimental Setup: Metrics

Model evaluation metrics

- Recall
- Precision
- F1-Score



Look Ahead Window (LAW)

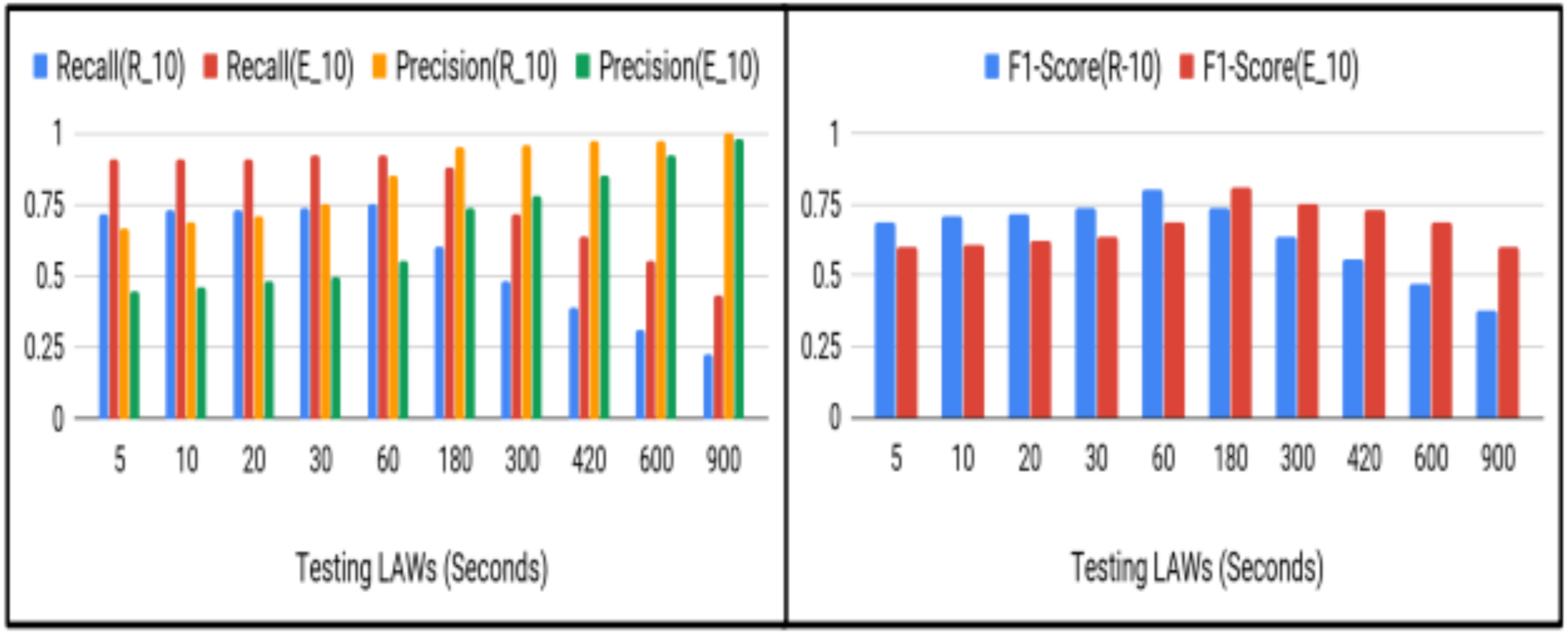


Experimental Setup: Configurations

Sets	S_1	S_2			S_3	
Models	***R_10***	***E_10***	***E_60***	***E_300***	***EE_10***	***EE_60***
Train LAWs	10 Seconds	10 Seconds	60 Seconds	300 Seconds	10 Seconds	60 Seconds
Train Applications	SensorSim	SensorSim + Surveillance			SensorSim + Surveillance + LPD	
Test Applications	Motion-Detector	Motion-Detector + SensorSim + LPD	Motion-Detector + LPD + Multitenancy	Motion-Detector	LPD	
Test Platform	Pi 0W	Pi 0W + Pi 3B + EC2	Pi 0W		Pi 3B	
Performance compared with	E_10	Threshold Tech., Compute Overhead	E_300	E_60	E_10	E_60

Results: Recall and Precision

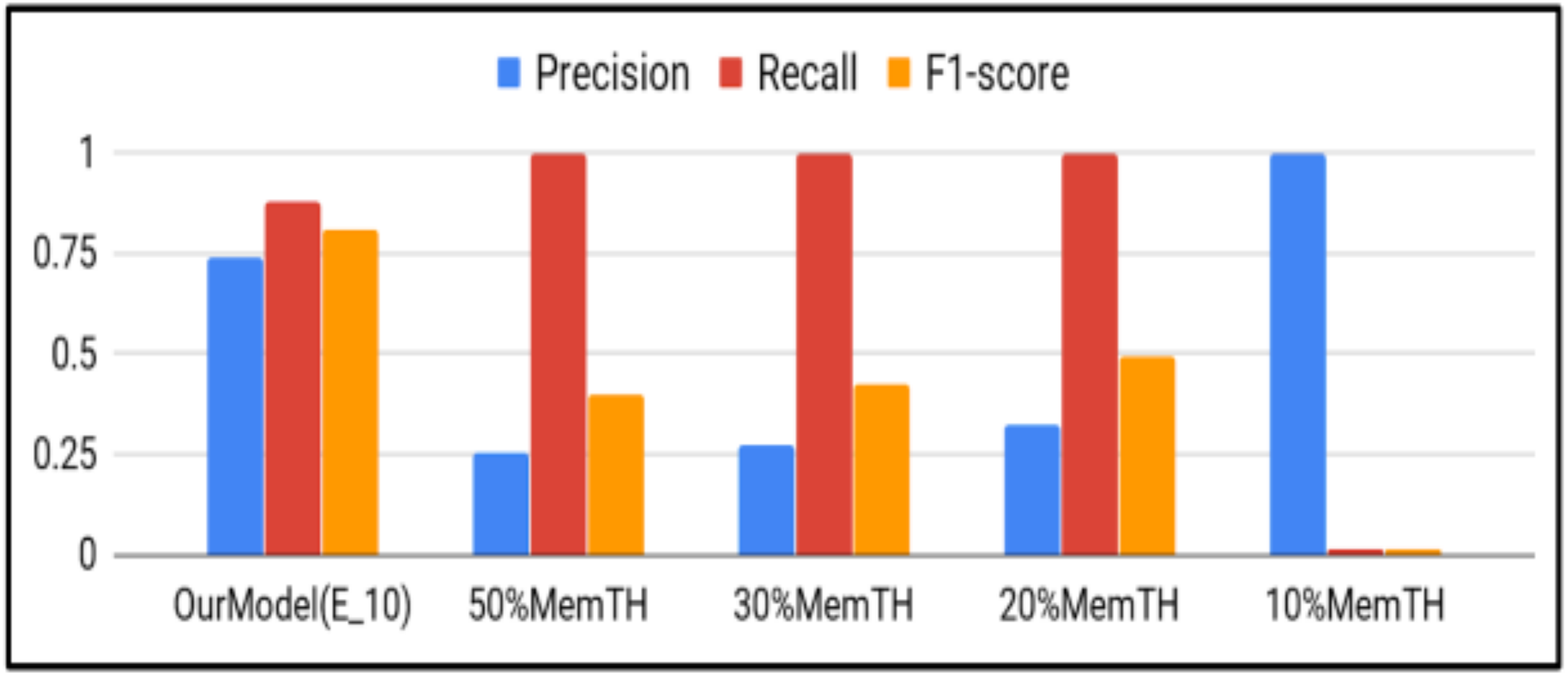
Surveillance on a Pi 0W



Recall rate is about 75% and precision is about 80% for a Look-ahead-Window (LAW) of 5 minutes (300 seconds)

Results: Threshold-based Schemes

Comparison with a Threshold-based system



Threshold-based schemes have higher recall but lower precision, or very low recall and high precision

Results: Overhead

Computational overhead		
Test data length in Seconds	Time Overhead in Seconds	Memory Overhead in MB
100	4.1	92.4
1000	5.1	93
5000	8.7	94.5
10300	13.1	96.9

Summary

- Memory exhaustion failures can be catastrophic in IoT devices
- Need failure prediction and mitigation techniques tuned for IoT
- Proposed MARK, that uses k-NN model for failure prediction
- Evaluated MARK under various real-world configurations.
  - MARK has precision and recall of over 75% with a 5 minute window