Resource- and Workload-aware Malware Detection through Distributed Computing in IoT Networks

By

Sreenitha Kasarapu, Sanket Shukla, Sai Manoj PD

Department of Electrical and Computer Engineering

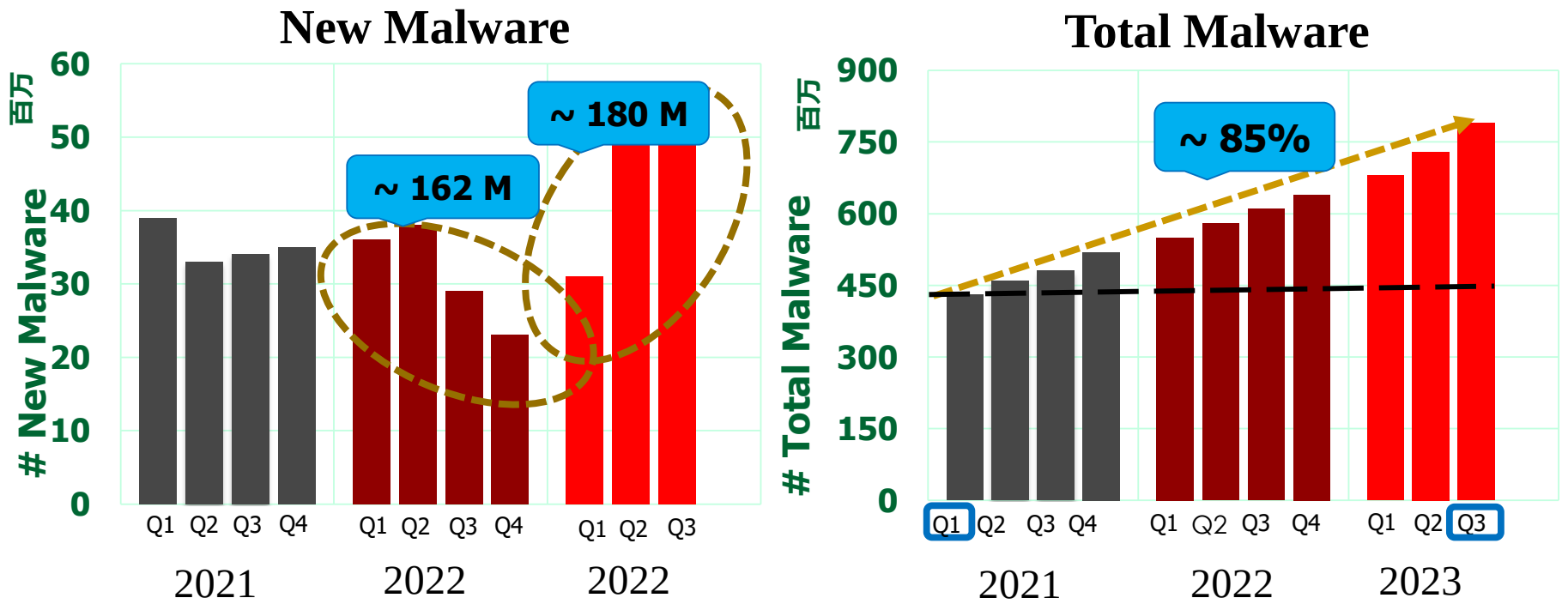
Email: {skasarap, sshukla4, spudukot}@gmu.edu

OUTLINE

- Introduction
- State-of-the-Art
- Proposed Technique
- Results
- Conclusion

INTRODUCTION

- Malware: Malicious software is created by an attacker to compromise security of a system or privacy of a victim.
- IoT devices are rapidly increasing and have become a target for numerous attacks



Source: McAfee Labs, November 2023

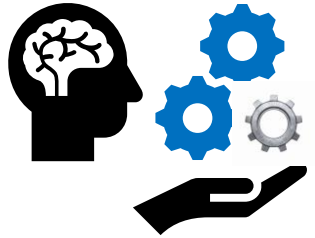
PROBLEM OVERVIEW

Reliable feature extraction

- Over-counting traits
- Misrepresenting system–level features

Real-time malware detection

- Need for minimal latency



Limited resources in IoT devices

- IoT devices designed for limited operations
- Portable and not equipped for inference task

RELATED WORK

Technique	Advantages	Disadvantages
Static analysis [1]	• Performed in a non-time environment • Simple • Computationally inexpensive [2]	• Remarkable overheads • Inefficient in detecting unseen threat [2] • Unreliable due to variant signatures of metamorphic, polymorphic and obfuscated malware. [2]
Dynamic Analysis [2] (code instruction changes, system calls, API calls, registry changes and memory writes)	• Functionality test • Efficient	

[1] A. Moser and et.al, “Limits of static analysis for malware detection,” in Annual Computer Security Applications Conference (ACSAC 2007), 2007.
[2] C. Rossow and et.al, “Prudent practices for designing malware experiments: Status quo and outlook,” Symposium on Security and Privacy, 2012.

RELATED WORK (Contd.)

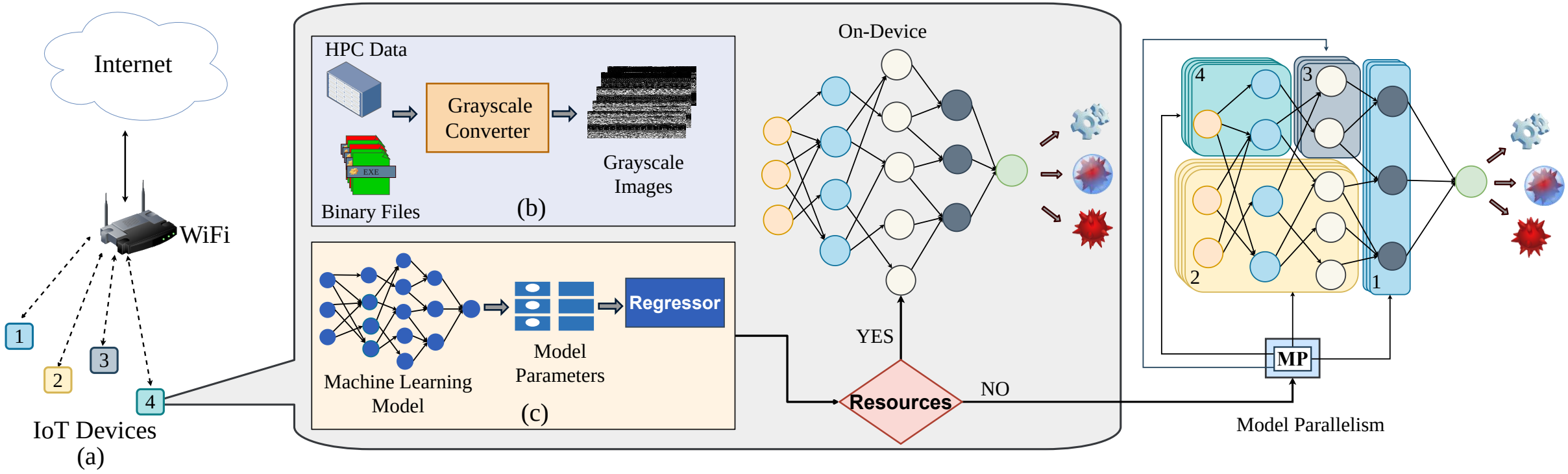
Technique	Advantages	Disadvantages
Image processing [3]	• Lesser time consumption because there is no need to dissemble feature maps nor observe the executable functionality like static and dynamic analysis	• Need to be trained with large amounts of training data to perform classification efficiently [3]
Model Parallelism [4]	• Suitable for training a massive ML model with limited resources by distributing the training process	• The communication delays between nodes could be costly

[3] A. Makandar and A. Patrot, “Malware class recognition using image processing techniques,” in Int. Conf. on Data Management, Analytics and Innovation (ICDMAI), 2017
[4] M. Shoeybi, M. M. A. Patwary, R. Puri et al., “Megatron-lm: Training multi-billion parameter language models using gpu model parallelism,”arXiv, 2019

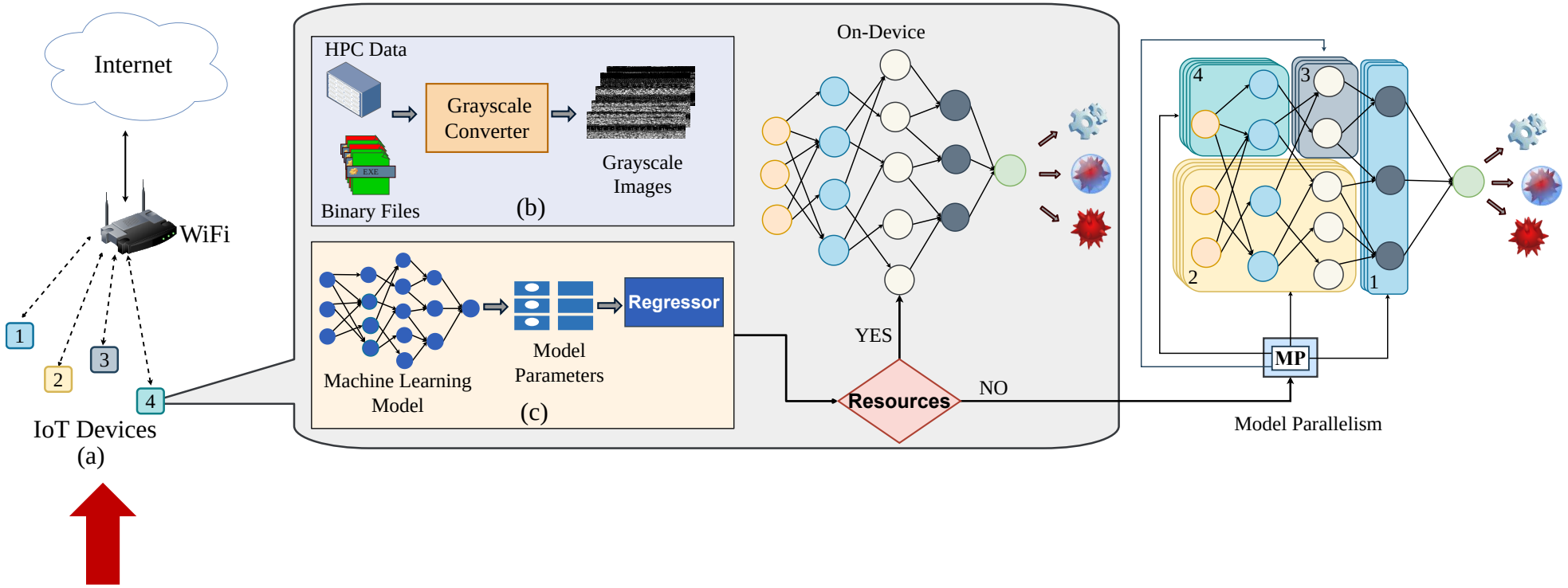
OVERVIEW OF THE PROPOSED TECHIQUE

- Introduces a resource- and workload-aware malware detection technique for resource constrained IoT devices.
- An automatic resource estimation is performed using a lightweight regression model to analyze the resources required for malware detection.
- Depending on the available resources, executing workloads, and communication overheads, malware detection (inference task) is either performed on-device or off-loaded to neighboring nodes with sufficient resources.

OVERVIEW OF THE PROPOSED TECHNIQUE (Contd.)



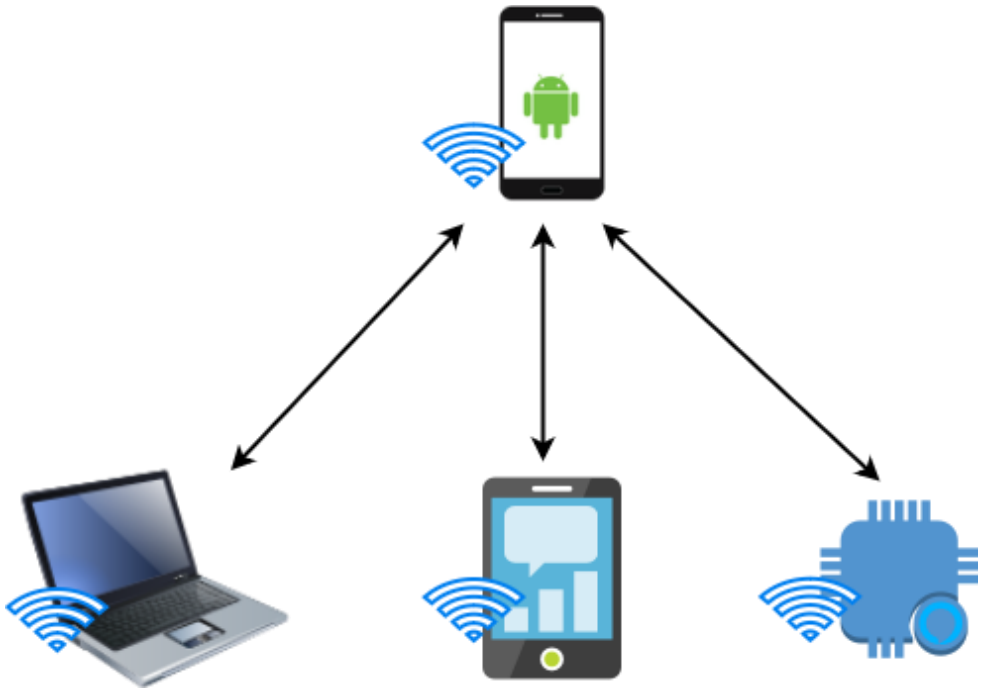
PROPOSED TECHNIQUE



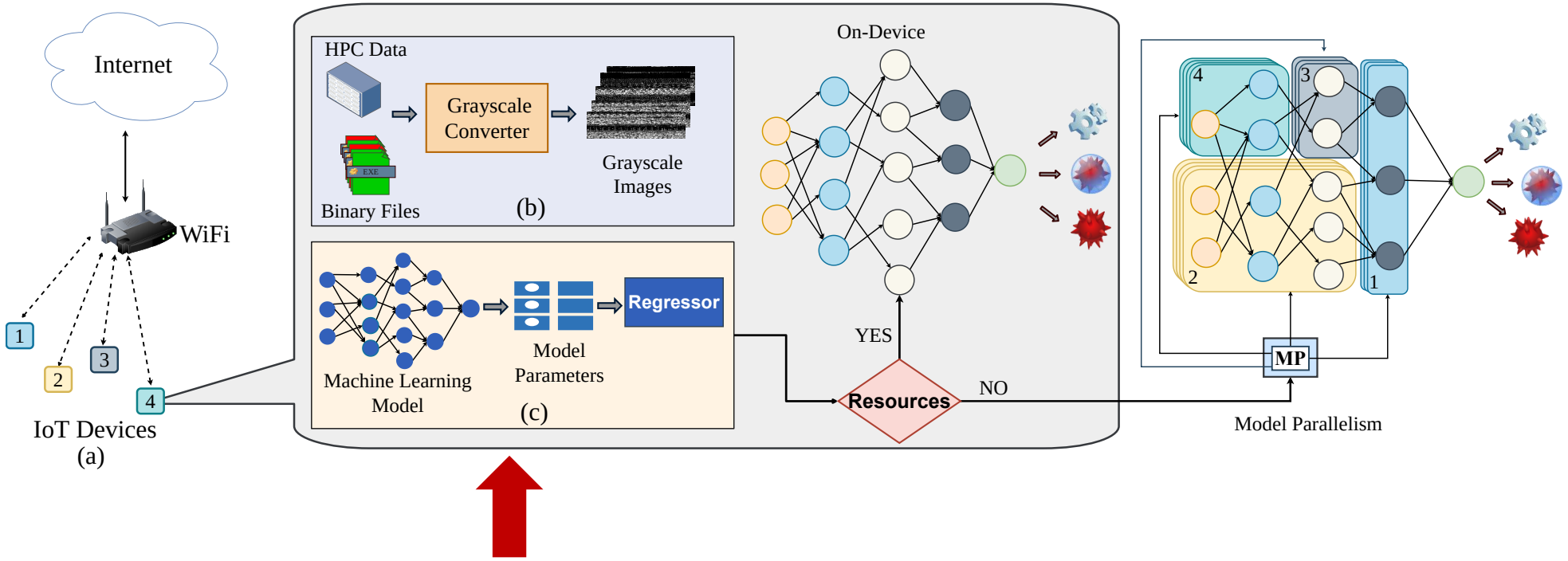
- Represents the IoT devices present in a network connected via WiFi.
- The resources from multiple devices are exploited to facilitate the malware inference task.

PROPOSED TECHNIQUE (Contd.)

- The different IoT devices present in a network.
- Sharing resources to facilitate inference task.
- It is assumed that there are no security threats, as the devices are within the same network.
- Also, the communication delays are negligible as the devices are in same network.



PROPOSED TECHNIQUE (Contd.)



- The microarchitectural event traces captured through HPCs are necessary for malware detection to address the reliability concerns.
- The resources needed to perform the inference task is calculated using the regressor.

PROPOSED TECHNIQUE (Contd.)

- A binary regression model is trained using data such as CNN's parameters, memory requirements of these parameters, and available memory at each node.
- The binary regression model gives an estimate of whether the CNN model inference can be performed on a single node or must be distributed onto multiple nodes.

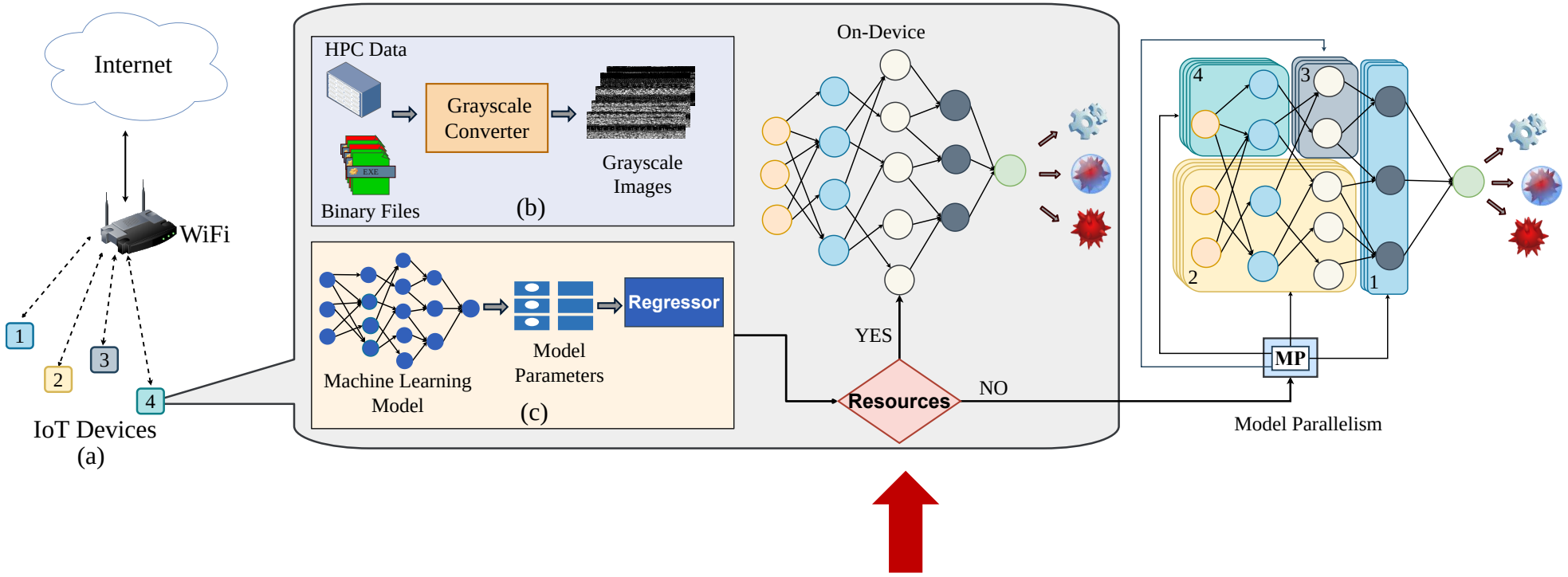
TABLE I
PARAMETER ESTIMATIONS PER EACH LAYER IN A CNN ALGORITHM

Layers	Description	Parameters
Input	No learnable parameters	0
CONV	(width of filter * height of filter * No. of filters in previous layer+1) * No. of filters in current layer	$f_{conv} = (w * h * p) + 1) * c$
POOL	No learnable parameters	0
FC	(current layer neurons * previous layer neurons)+1 * current layer neurons	$f_{FC} = (n_c * n_p) + 1 * n_c$
Softmax	(current layer neurons * previous layer neurons) + 1 * current layer neurons	$f_S = (n_c * n_p) + 1 * n_c$

Algorithm 2 Lightweight Linear Regression Algorithm

Require: B_{exe} (Benign application files), M_{exe} (Malware application files)
Input: $\mathfrak{Mem}[node], \mathbb{C}$
  define $Regressor(\mathbb{C})$:
    for $layer \leftarrow \mathbb{C}$: **do**
      $var \leftarrow f(W, B, A)$
      if $layer \rightarrow CONV$
        $par = f_{conv}(var)$
      elif $layer \rightarrow (FC \vee Softmax)$
        $par = f_{FC}(var)$
      else
        $par = 0$
      end if
      $\bar{(P)}.append(par)$
    end for
    $\mathfrak{Mem}[model] \leftarrow N * batch\_size * \bar{(P)} * 1KB$
    $X_R.features \leftarrow \{W, A, B, \bar{(P)}, \mathfrak{Mem}[model],$
                $\mathfrak{Mem}[node]\}$
    $Res \leftarrow \mathbb{R} : (X_R, \beta)$
    return Res

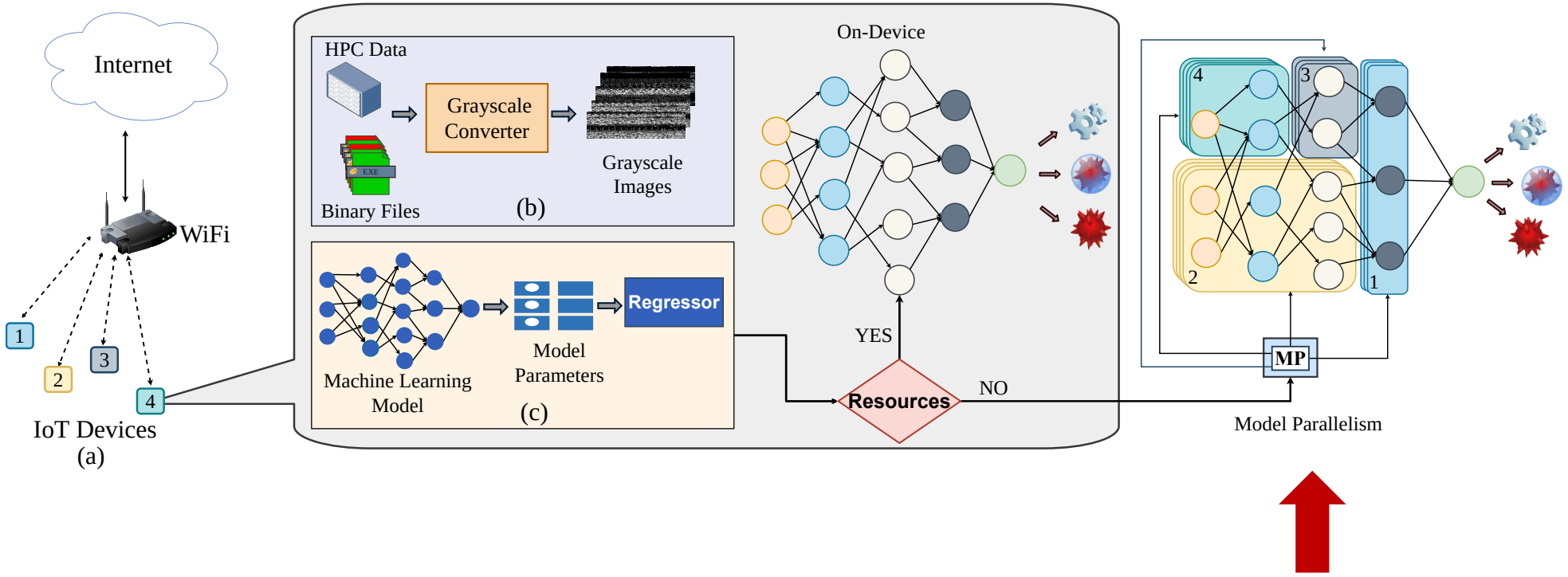
PROPOSED TECHNIQUE (Contd.)



- The resources required to perform the inference task are compared with the available resources.
- If the inference task fails to follow the condition, the task is divided amongst multiple nodes.

$$\mathfrak{Mem}[model] \leq \mathfrak{Mem}[node]$$

PROPOSED TECHNIQUE (Contd.)



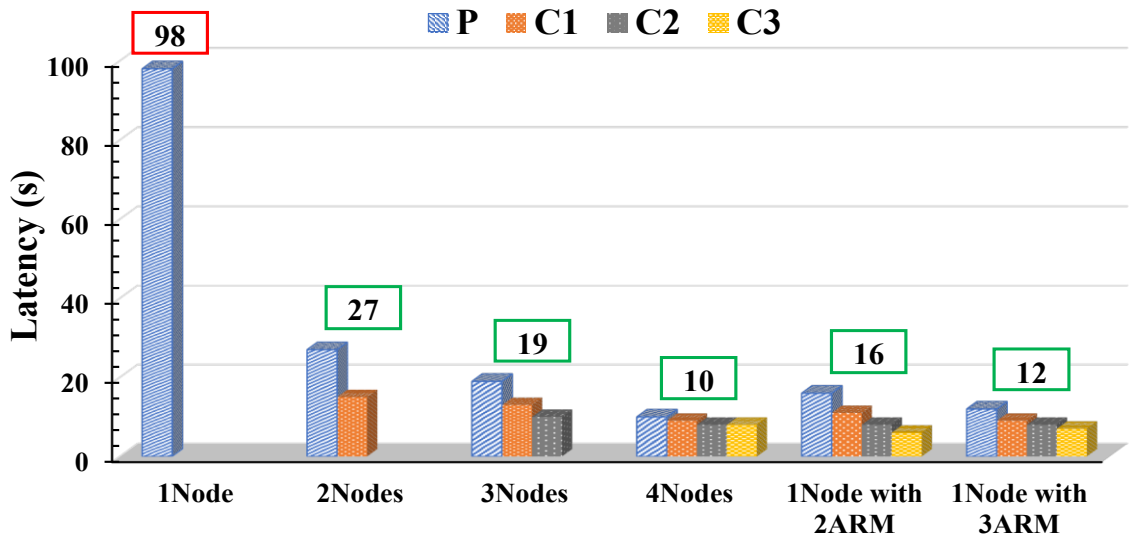
- When there are no enough resources, the inference task is divided on to multiple nodes.
- The parent node is responsible for data flow (gradient updates) between child nodes.

EXPERIMENTAL SETUP

- Collected 70,000 malware samples that encompass 5 malware classes: backdoor, rootkit, trojan, virus, and worm.
- Collected 12,500 benign application files.
- Deployed 20 IoT nodes encompassing Broadcom BCM2711, Jeston Nanos, and quad- core Cortex-A72 (ARM v8) 64-bit boards. (Which represent IoT devices in real-time connected via WiFi)
- The deployed Jetson Nanos contains a 128-core NVIDIA Maxwell architecture-based GPU and Quad-core ARM® A57 CPU.

EXPERIMENTAL RESULTS (Contd.)

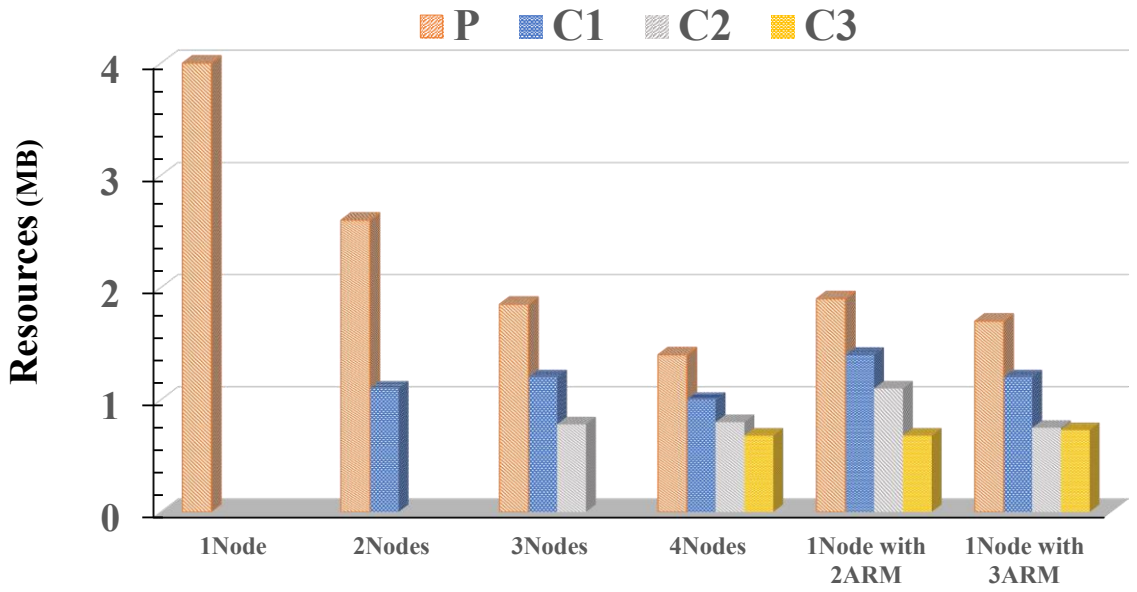
- Normalized inference execution time is analyzed for cases: a) the parent node has sufficient resources; b) the parent node does not have enough resources and outsources to multiple nodes.
- For the case of sufficient resources, it takes 98 seconds to perform the inference task.
- For the case of model parallelism, we can observe a speedup of 4× when the inference task is parallelized between two nodes.



Massive speedups of up to to 9.8 x were observed with the proposed resource and workload-aware distributed learning technique.

EXPERIMENTAL RESULTS (Contd.)

- The inference task takes 4 MB of data to complete.
- In the first case, the single parent node P can provide this data to complete the inference task.
- In other cases, the inference task is divided between multiple nodes (model parallelism), so the data required is also divided into multiple nodes.



Distributing the inference task on multiple nodes reduces the resource exhaustion in IoT devices by more than 50%.

EXPERIMENTAL RESULTS (Contd.)

- Compared to the existing techniques, the proposed resource-aware CNN-based distributed training on HPC-based image data achieves the highest accuracy. It maintains an average accuracy of 96.7%.

TABLE I
COMPARISON WITH EXISTING HPC-BASED DETECTION TECHNIQUES

Model	Accuracy (%)	F1-score	Recall
OneR [21]	0.81	0.81	0.82
JRIP [21]	0.83	0.83	0.84
PART [21]	0.81	0.815	0.831
J48 [21]	0.82	0.82	0.82
Adaptive-HMD [22]	0.853	0.853	0.858
SVM [23]	0.739	0.736	0.772
RF [23]	0.835	0.834	0.822
NN [23]	0.811	0.811	0.816
SMO [24]	0.932	0.933	0.931
Proposed (MP)	**0.967**	**0.967**	**0.972**

The proposed technique reduces the inference latency with minimal resource consumption along with retaining the high malware detection capability of 96.7% in IoT devices.

CONCLUSIONS

- We addressed the problem of malware detection in IoT nodes which are highly resource constrained.
- With the proposed resource- and workload-aware model parallelism-based malware detection technique which employs distributed inference, better security for resource-constrained IoT devices is enabled.
- A robust HPC collection mechanism was introduced to address the concerns of reliability of training data.
- The proposed technique can achieve a speed-up of 9.8× compared to on-device inference while maintaining a malware detection accuracy of 96.7%.

ACKNOWLEDGEMENT

- This work was supported by the Commonwealth Cyber Initiative, and investment in the advancement of cyber R&D innovation, and workforce development. For more information about CCI, visit www.cyberinitiative.org

THANK YOU