Dependency and Priority Based Multi-Queue IoT Task Scheduling in Heterogeneous Systems

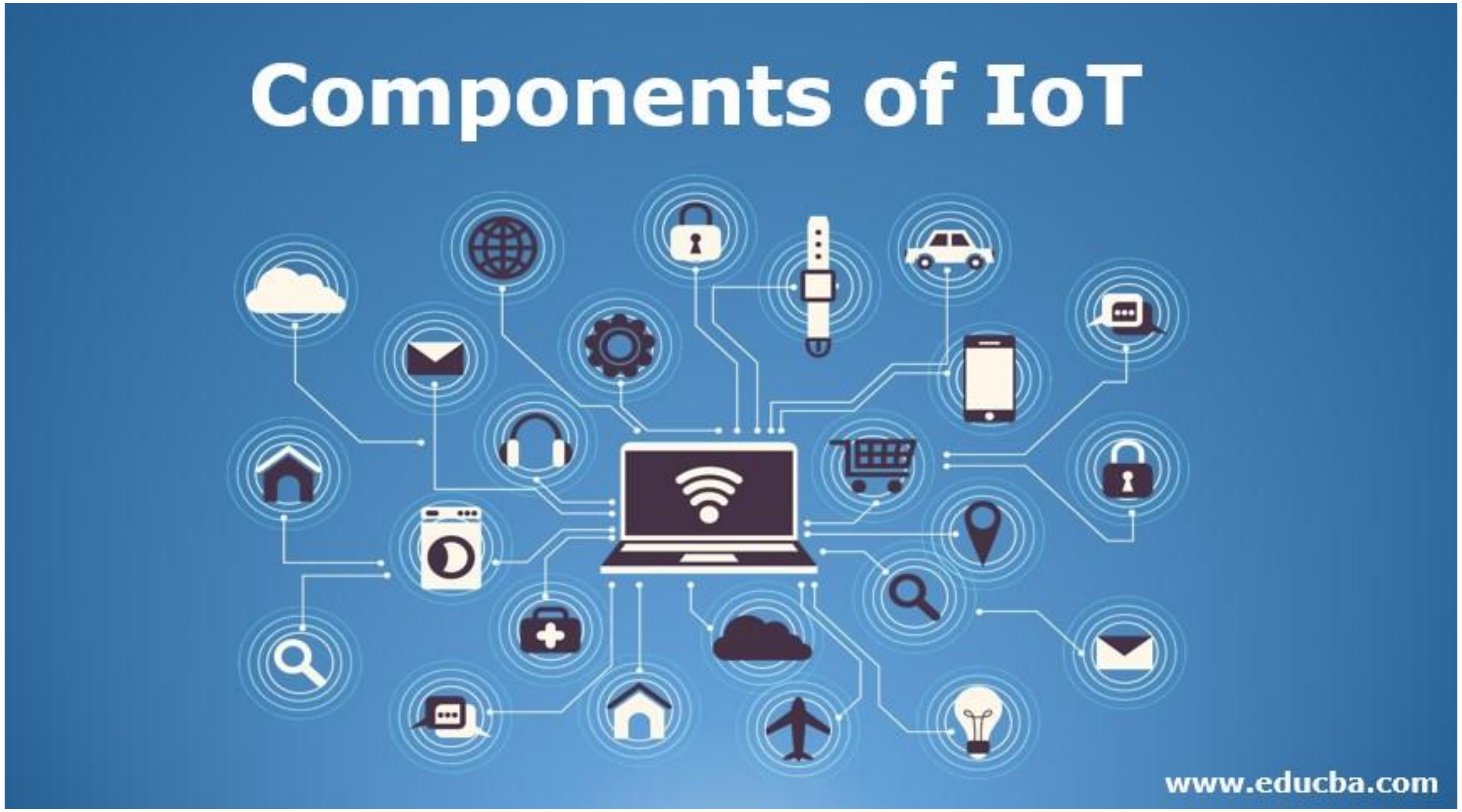
Supervised By:
Dr. Rezwana Reaz
Assistant Professor
Department of CSE, BUET

Presented By:
Muhammad Ehsanul Kader (1805067)
Maneesha Rani Saha (1805076)

OUTLINE

PRELIMINARIES

INTERNET OF THINGS



A network of physical objects embedded with sensors, software, and other technologies to collect and exchange data over the internet.

TASK

A unit of work that–

- Needs to be completed within a specified timeframe
- Using a finite set of resources
- Involves complex management and processing of large datasets

TASK SCHEDULING

Process of allocating computational tasks to appropriate resources with a goal to –

- Optimize performance
- Minimize response time
- Reduce cost and energy

LIMITATIONS OF IoT DEVICES

- Low computing resources
- Low energy resources
- Limited storage capacity
- Low data rate
- Stringent chip area

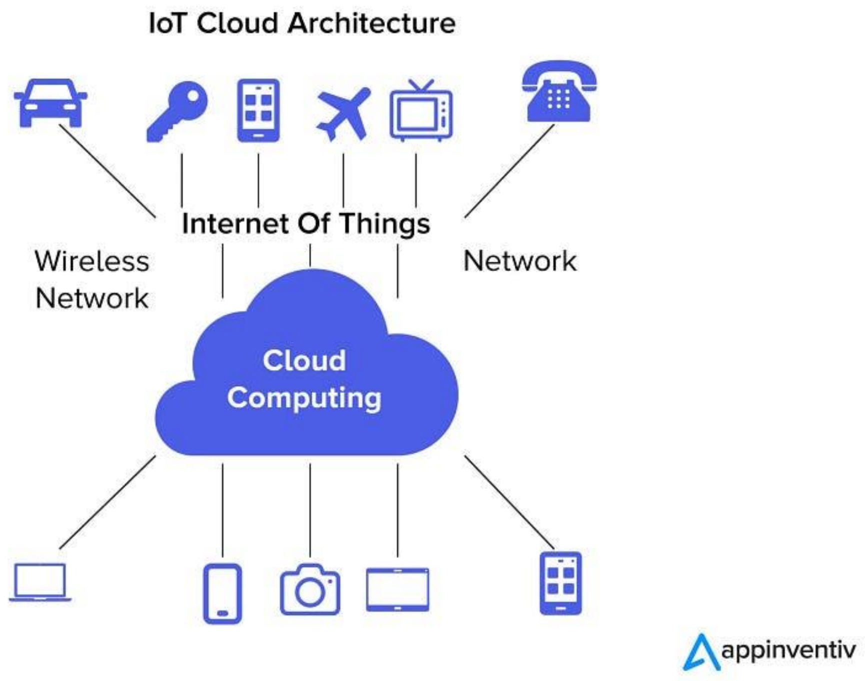
CLOUD COMPUTING

Advantage:

- Can process and analyze the enormous amounts of data produced by IoT devices.

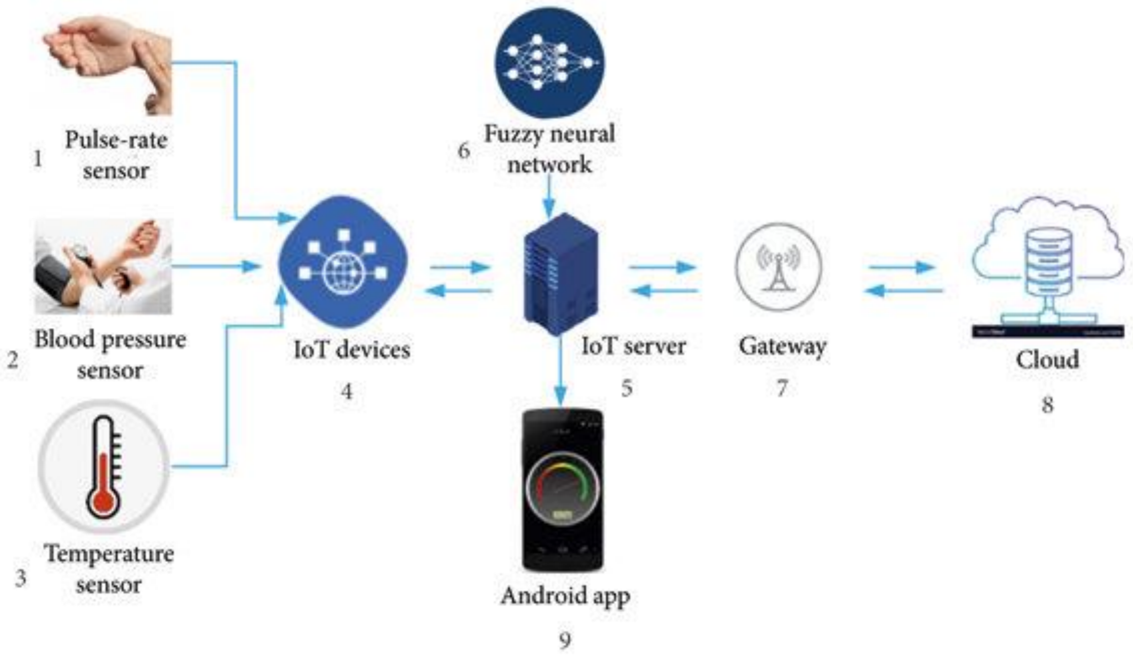
Disadvantage:

- High latency
- Low scalability
- Centralized distribution



DELAY SENSITIVE IoT APPLICATIONS

- Designed to operate within specific time constraints
- Any delays or variations in timing affects their performance or functionality

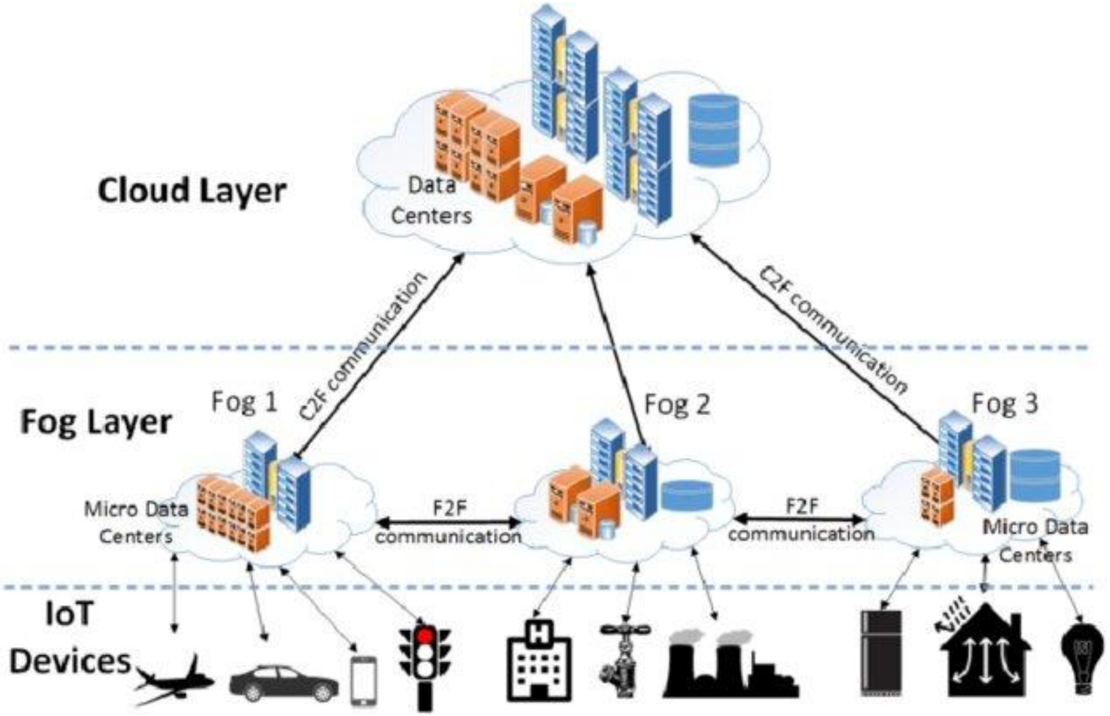


Example: IoT in Healthcare

FOG COMPUTING

Fog Layer:

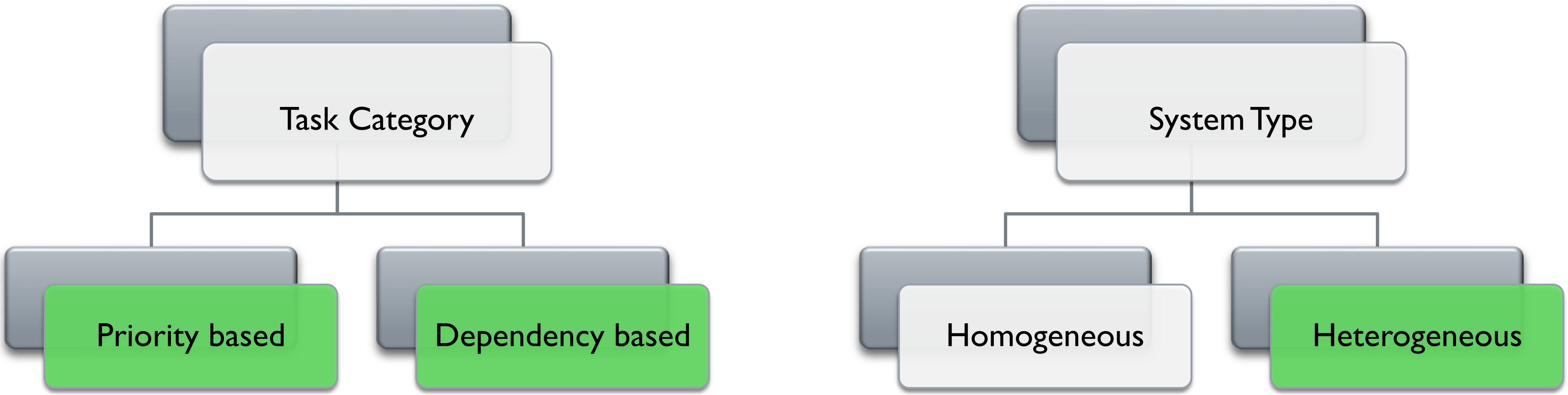
- An decentralized intermediate layer between the IoT devices and the cloud.
- Reduces communication delays compared to cloud-only architecture.
- Enables the implementation of many latency-sensitive IoT services.
- Lower storage and processing capabilities than the cloud



IoT TASK WORK CATEGORIES

- Task processing location / Architecture
  - Cloud
  - Fog-cloud
- Fog Architecture
  - Resource Allocation
  - Scheduling Algorithm

IoT TASK WORK CATEGORIES



IoT TASK WORK CATEGORIES

DOMAIN FOR OUR WORK

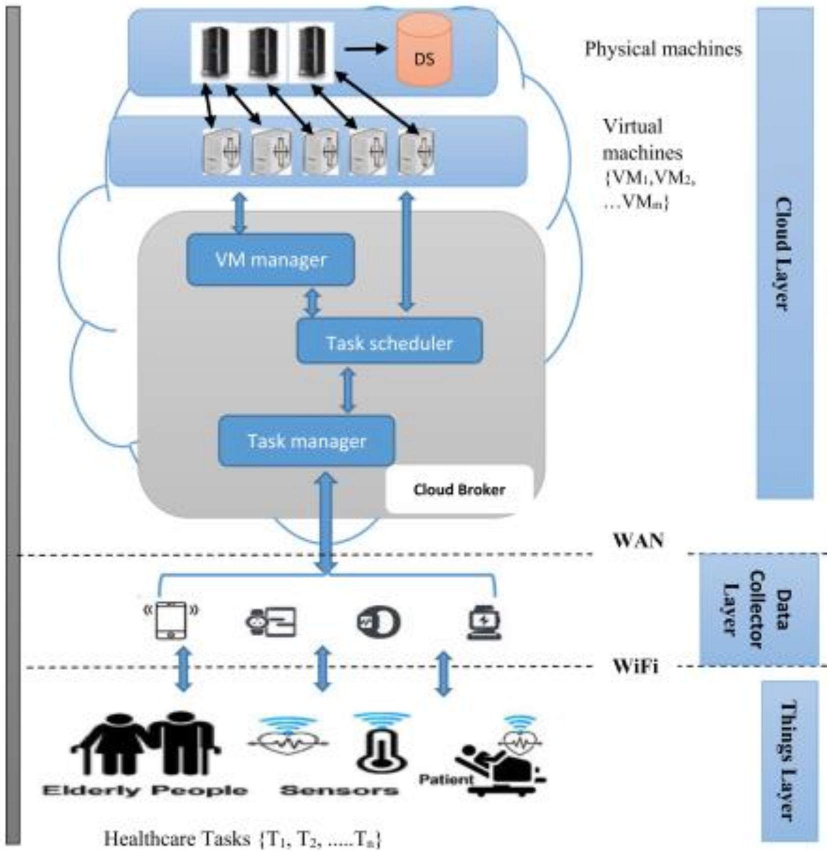
- Dynamic Batch Scheduling
- Priority and Dependency Based Task
- Heterogeneous System
- Fog Computing

LITERATURE REVIEW

PRIORITIZED SCHEDULING TECHNIQUE FOR HEALTHCARE TASKS IN CLOUD COMPUTING [1]

Key Contributions:

- Prioritized Sorted Task-Based Allocation Technique
- Scheduling occurs in **cloud layer**
- Tasks are first assigned into **multi-queue** depending on their priority value
- Task with higher priority and higher length is allocated to higher capacity VM **(heterogeneous)**



Cloud (Three layer) Architecture

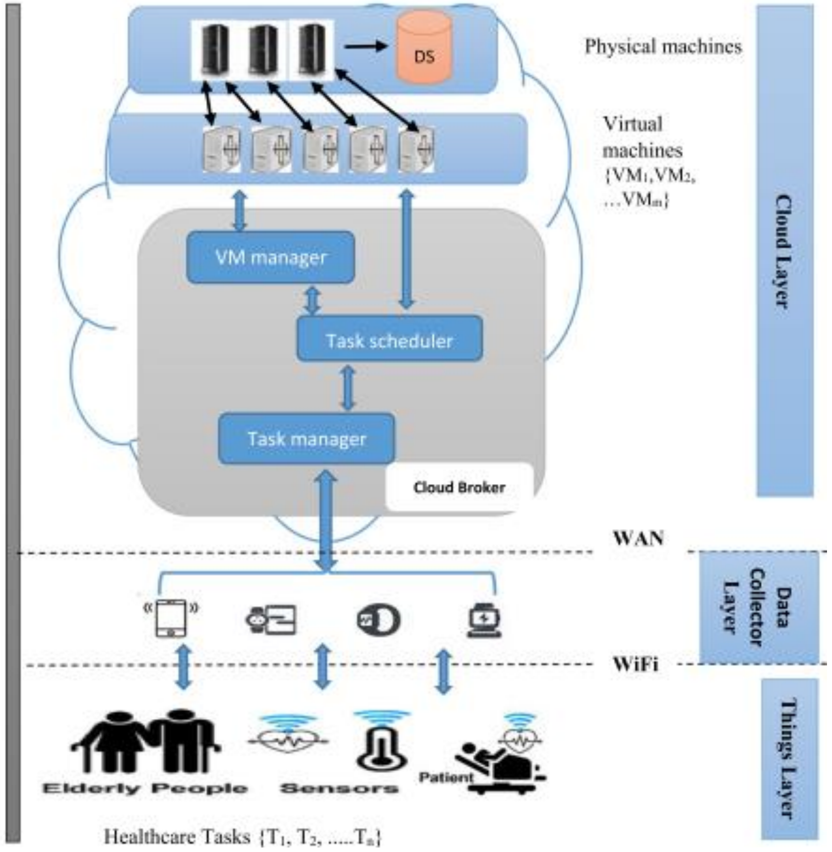
PRIORITIZED SCHEDULING TECHNIQUE FOR HEALTHCARE TASKS IN CLOUD COMPUTING [1]

Key Contributions:

- Prioritized Sorted Task-Based Allocation Technique
- Scheduling occurs in **cloud layer**
- Tasks are first assigned into **multi-queue** depending on their priority value
- Task with higher priority and higher length is allocated to higher capacity VM **(heterogeneous)**

Limitations:

- Did not consider **Fog**
- Did not consider **Task dependency**

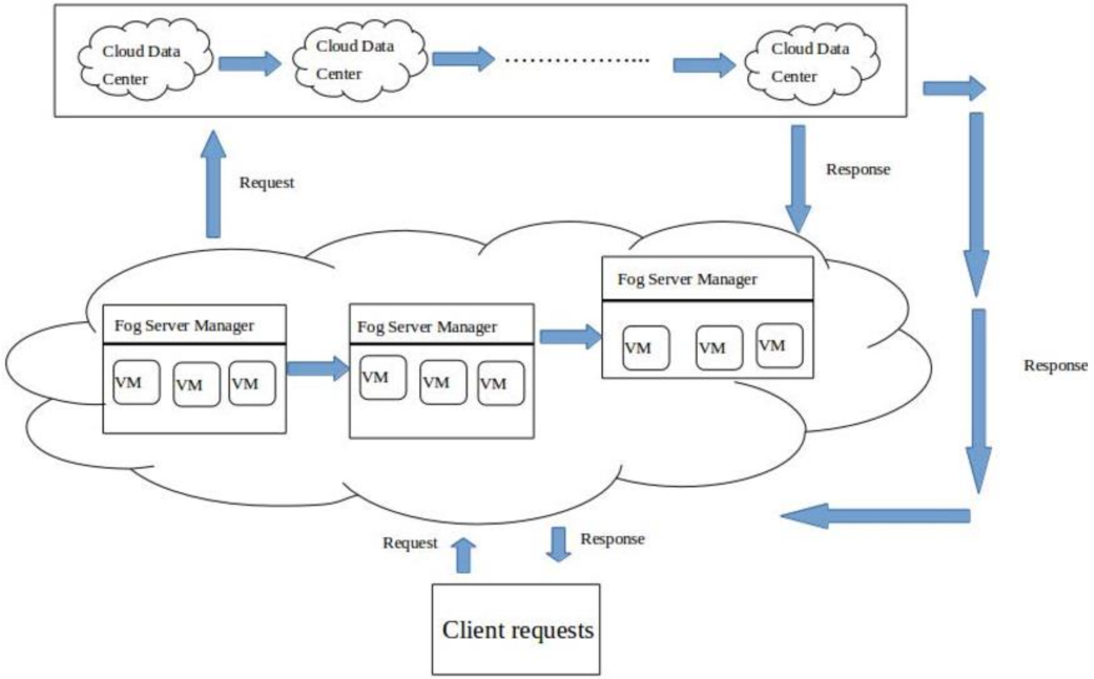


Cloud (Three layer) Architecture

PRIORITIZED TASK SCHEDULING IN FOG COMPUTING [2]

Key Contributions:

- Priority based **multi-queue** task scheduling algorithm
- Scheduling occurs in **fog layer**
- Assigns task to nearest fog node
- Task is sent to the cloud if the fog layer does not have sufficient resource



Fog-Cloud (Three layer) Architecture

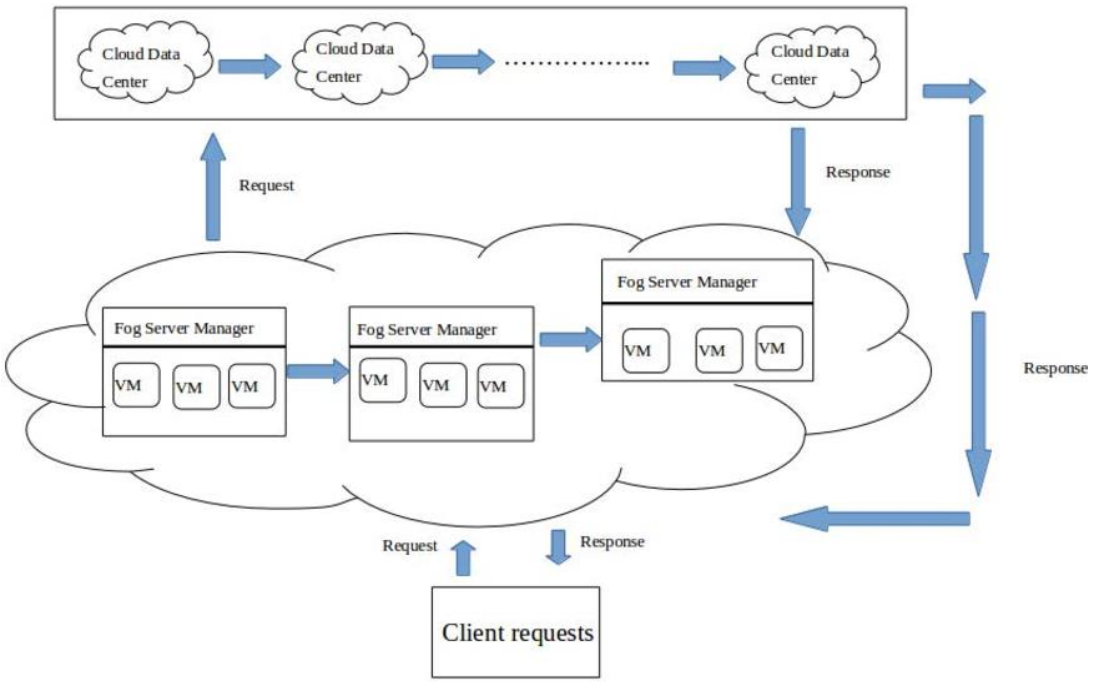
PRIORITIZED TASK SCHEDULING IN FOG COMPUTING [2]

Key Contributions:

- Priority based **multi-queue** task scheduling algorithm
- Scheduling occurs in **fog layer**
- Assigns task to nearest fog node
- Task is sent to the cloud if the fog layer does not have sufficient resource

Limitations:

- Did not consider **Task dependency**

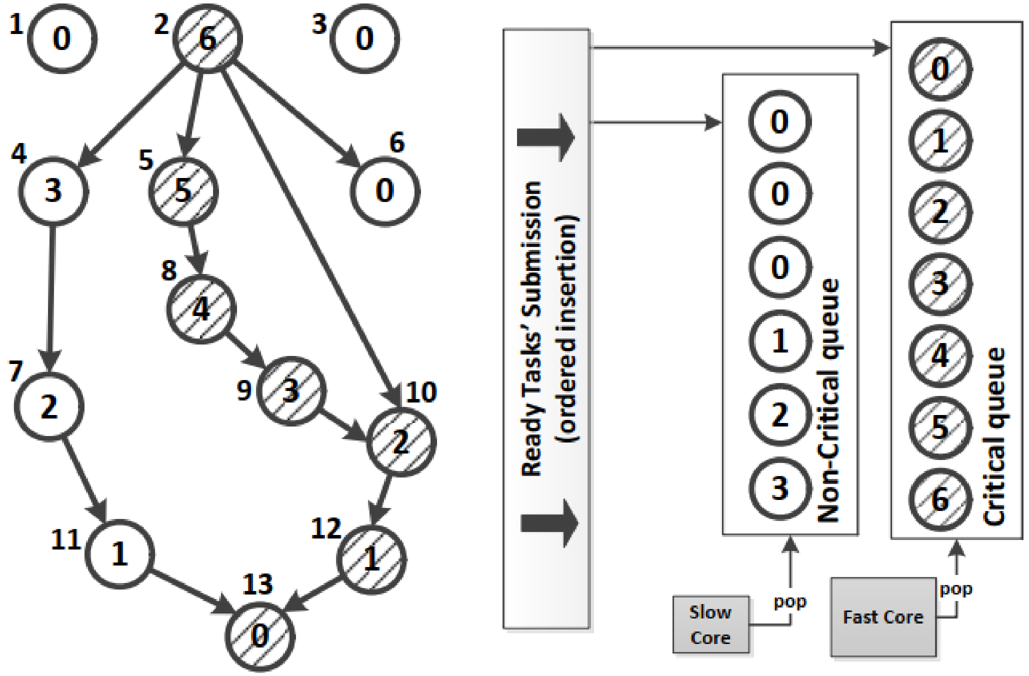


Fog-Cloud (Three layer) Architecture

CRITICALITY-AWARE DYNAMIC TASK SCHEDULING FOR HETEROGENEOUS ARCHITECTURES [3]

Key Contributions:

- Criticality-aware **dynamic** task scheduler (CATS)
- **Heterogeneous multi-core** (slow and fast) platform
- Tasks are considered critical if they are part of the longest path in the in-flight dynamic state of the dependency graph.



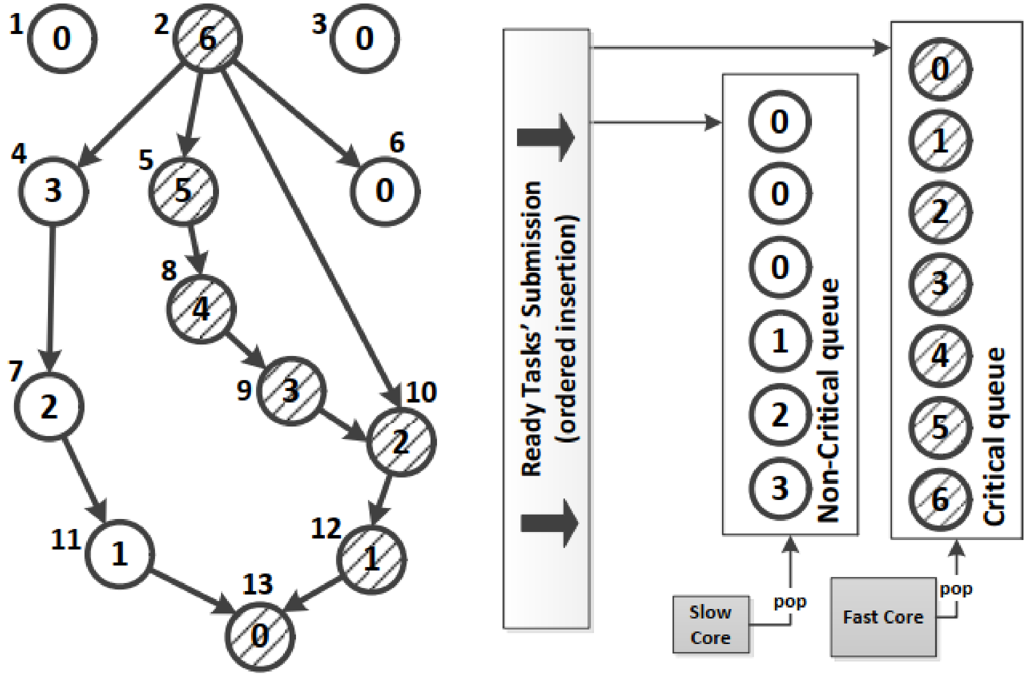
CRITICALITY-AWARE DYNAMIC TASK SCHEDULING FOR HETEROGENEOUS ARCHITECTURES [3]

Key Contributions:

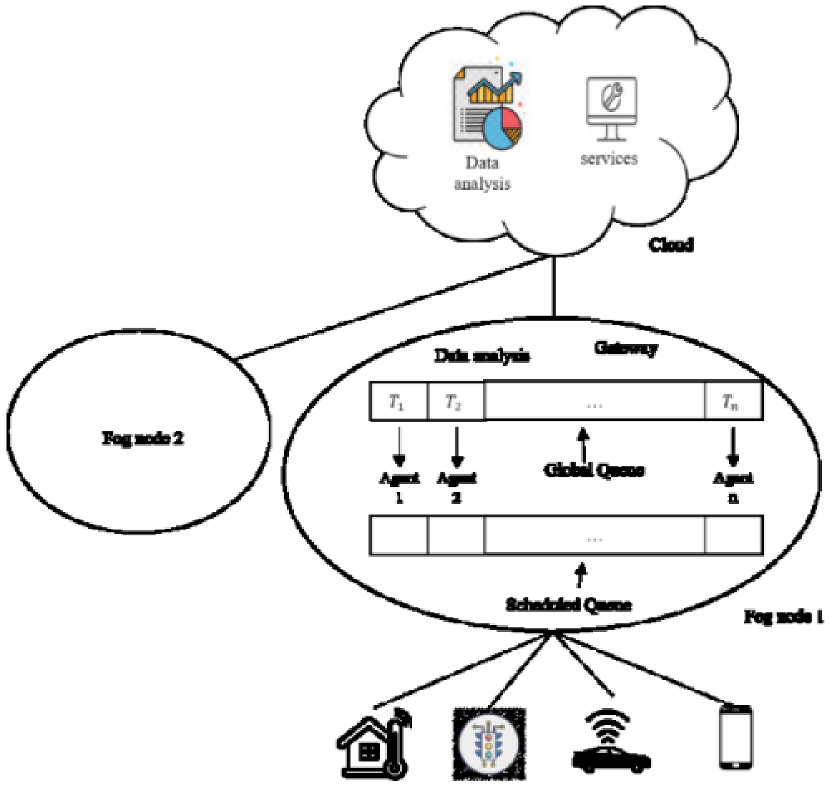
- Criticality-aware **dynamic** task scheduler (CATS)
- **Heterogeneous multi-core** (slow and fast) platform
- Tasks are considered critical if they are part of the longest path in the in-flight dynamic state of the dependency graph.

Limitations:

- Only one dependency chain is given priority



A MULTI-AGENT BASED MODEL FOR TASK SCHEDULING IN CLOUD-FOG COMPUTING PLATFORM [4]



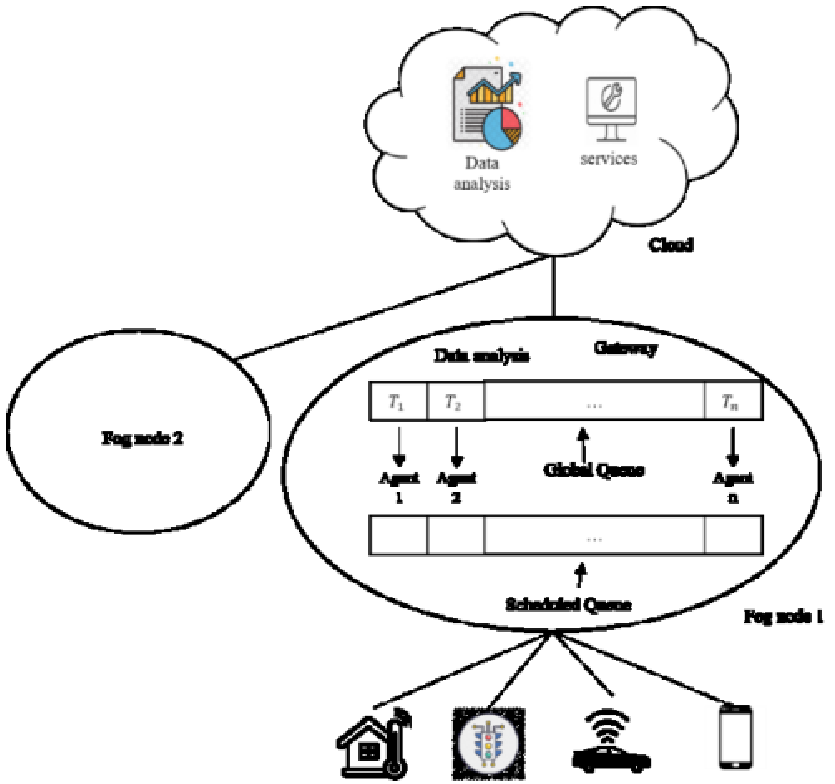
Key Contributions:

- Multi-agent based **non-preemptive** task scheduler
- Scheduling occurs in **fog layer**
- **Single-queue** implementation
- Tasks are sorted according to their importance values (waiting time, status and priority values of dependent tasks) and also the number of resources required to be performed

A MULTI-AGENT BASED MODEL FOR TASK SCHEDULING IN CLOUD-FOG COMPUTING PLATFORM [4]

Key Contributions:

- Multi-agent based **non-preemptive** task scheduler
- Scheduling occurs in **fog layer**
- **Single-queue** implementation
- Tasks are sorted according to their importance values (waiting time, status and priority values of dependent tasks) and also the number of resources required to be performed

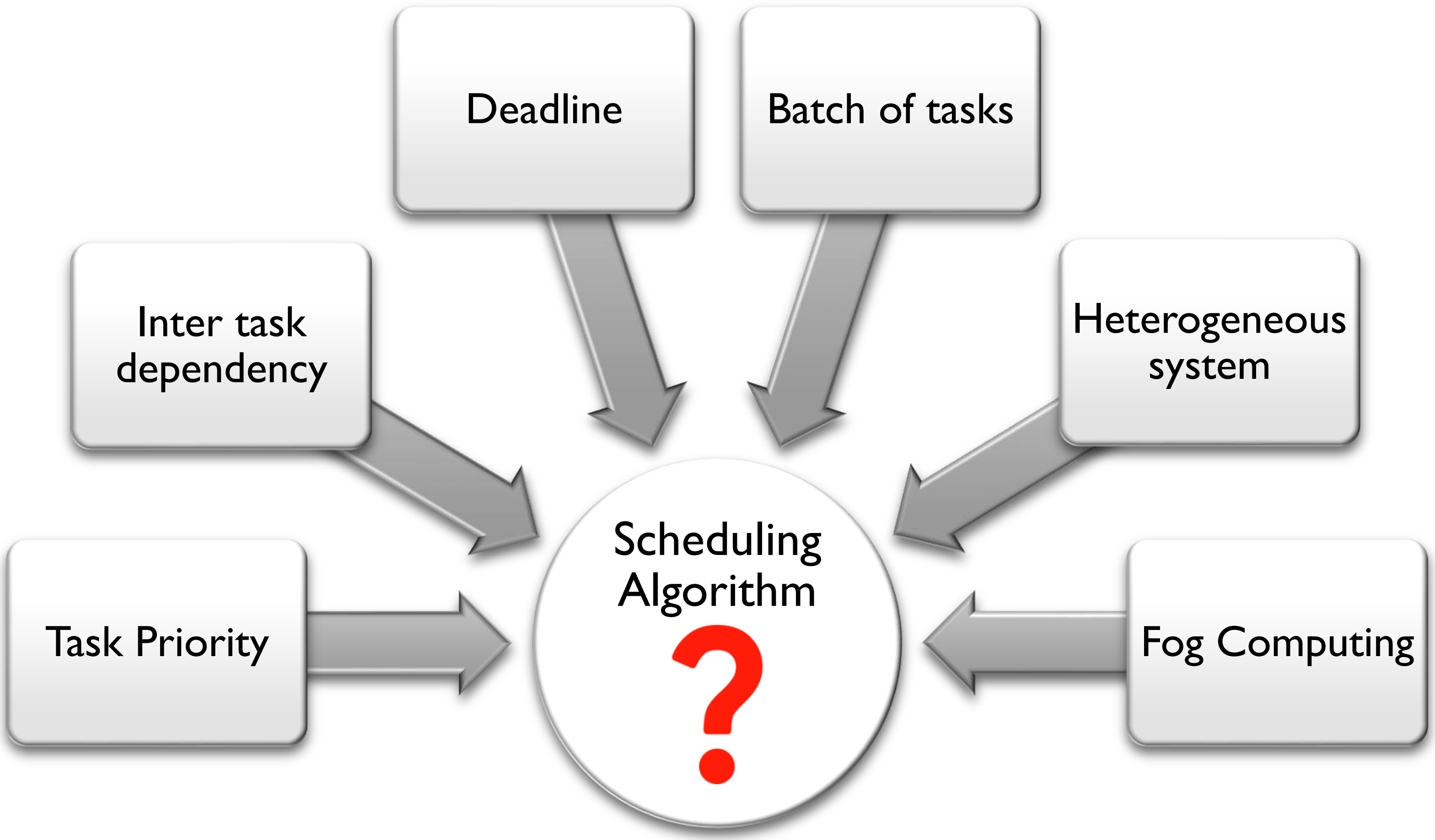


Limitations:

- Did not consider **Multi-queue**
- **Single VM System**

MOTIVATION

MOTIVATION


Deadline
Batch of tasks
Inter task dependency
Heterogeneous system
Task Priority
Scheduling Algorithm
?
Fog Computing

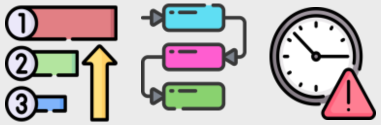
PROBLEM DEFINITION

PROBLEM DEFINITION

Design and Implementation of an **IoT Task Scheduling Algorithm** that -

PROBLEM DEFINITION

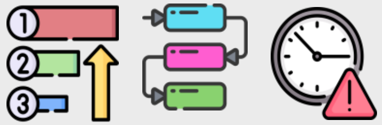
Design and Implementation of an **IoT Task Scheduling Algorithm** that -



Considers **Task Priority, Inter-Dependency and Deadline**

PROBLEM DEFINITION

Design and Implementation of an **IoT Task Scheduling Algorithm** that -



Considers **Task Priority, Inter-Dependency and Deadline**

Considers **Batch of Tasks for Dynamic Resource Assignment**

PROBLEM DEFINITION

Design and Implementation of an **IoT Task Scheduling Algorithm** that -

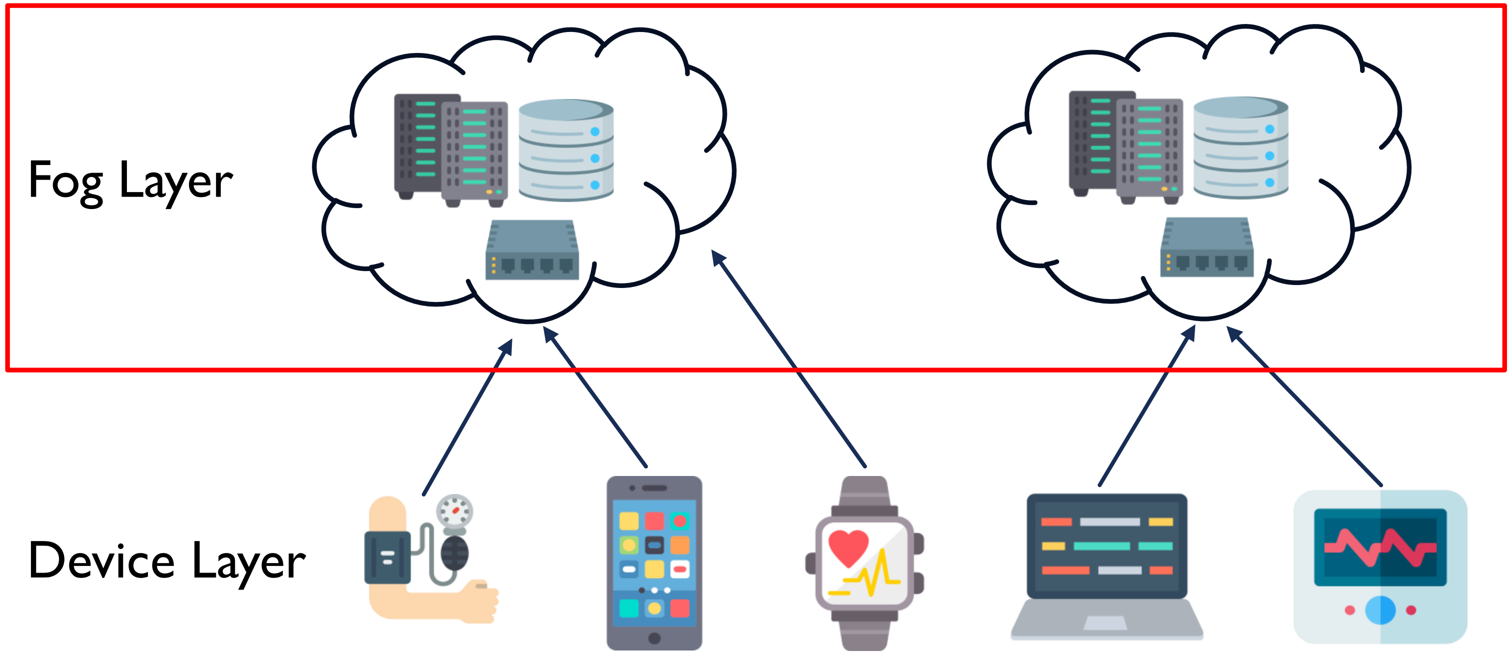
- Considers **Task Priority, Inter-Dependency and Deadline**
- Considers **Batch of Tasks for Dynamic Resource Assignment**
- Introduces a **Numerical Scoring Mechanism** incorporating task priority and dependency

OUR CONTRIBUTION

- Designing and Implementing a **Dynamic Batch Scheduling Algorithm** for **Delay-Sensitive** IoT Tasks with Varying **Priorities and Inter-task Dependencies**
- Introducing a **Numerical Scoring Mechanism** for Every Task Incorporating Task Priority and Dependencies
- Utilizing **Multi Queues** in **Heterogeneous Fog Computing System**

METHODOLOGY

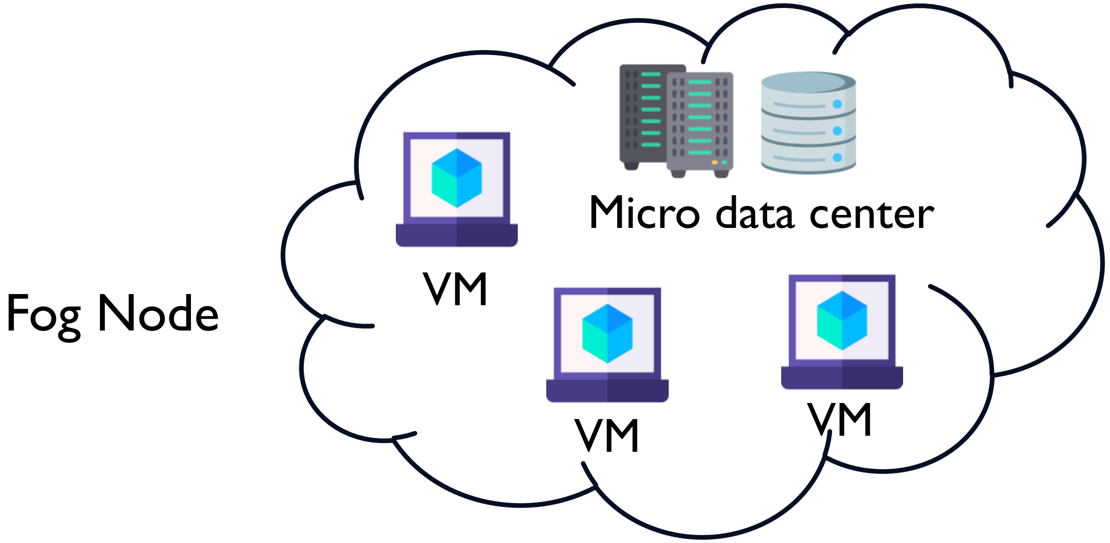
SYSTEM ARCHITECTURE



FOG LAYER

- Consists of several Fog Nodes.
- Task from IoT devices are sent to nearest Fog node.

FOG NODE



Each fog node is composed of –

- a micro data center
- Several heterogeneous VMs

TASK GENERATION

We have generated a **Synthetic Taskset** for our purpose that has the following properties –

- Task ID
- Priority Level (High=1, Medium=2, Low=3)
- MI (Million Instructions referred from Benchmark GoCJ_Dataset)
- Deadline (Process Time+Delay Time)
- Predecessor Task List

TASK GENERATION

We have generated a **Synthetic Taskset** for our purpose that has the following **hyperparameters** –

- Number of Total Tasks in a Batch
- Dependent Task Ratio
- Task Priority Level Ratio $[w_H + w_M + w_L = 1]$
- Number of Predecessor for a Task (min, max)

ALGORITHM OVERVIEW

A fog node receives a batch of tasks

↓

Task priority is updated if needed

↓

Priority-Dependency score is calculated for every task

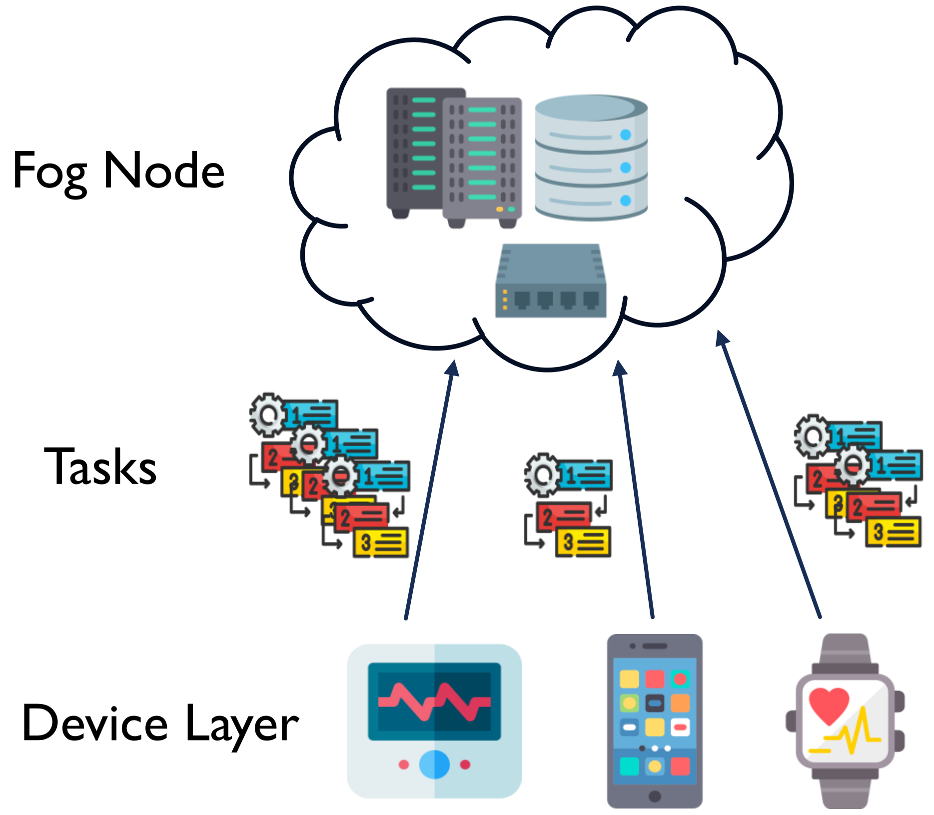
↓

Tasks are inserted into respective high, mid and low priority level queues (waiting and ready)

↓

Task is selected from ready queue based on deadline and PD_score and sent to the respective VM for execution

BATCH OF TASK ARRIVAL



A fog node receives a batch of tasks

↓

Task priority is updated if needed

↓

Priority-Dependency score is calculated for every task

↓

Tasks are inserted into respective high, mid and low priority level queues (waiting and ready)

↓

Task is selected from ready queue based on deadline and PD_score and sent to the respective VM for execution

TASK PRIORITY UPDATE

Task Priority

- Original Priority
- Imposed Priority

Why Imposed Priority?

Prevent lower priority parent from causing higher priority child to miss deadline

A fog node receives a batch of tasks

↓

Task priority is updated if needed

↓

Priority-Dependency score is calculated for every task

↓

Tasks are inserted into respective high, mid and low priority level queues (waiting and ready)

↓

Task is selected from ready queue based on deadline and PD_score and sent to the respective VM for execution

UPDATE TASK PRIORITY

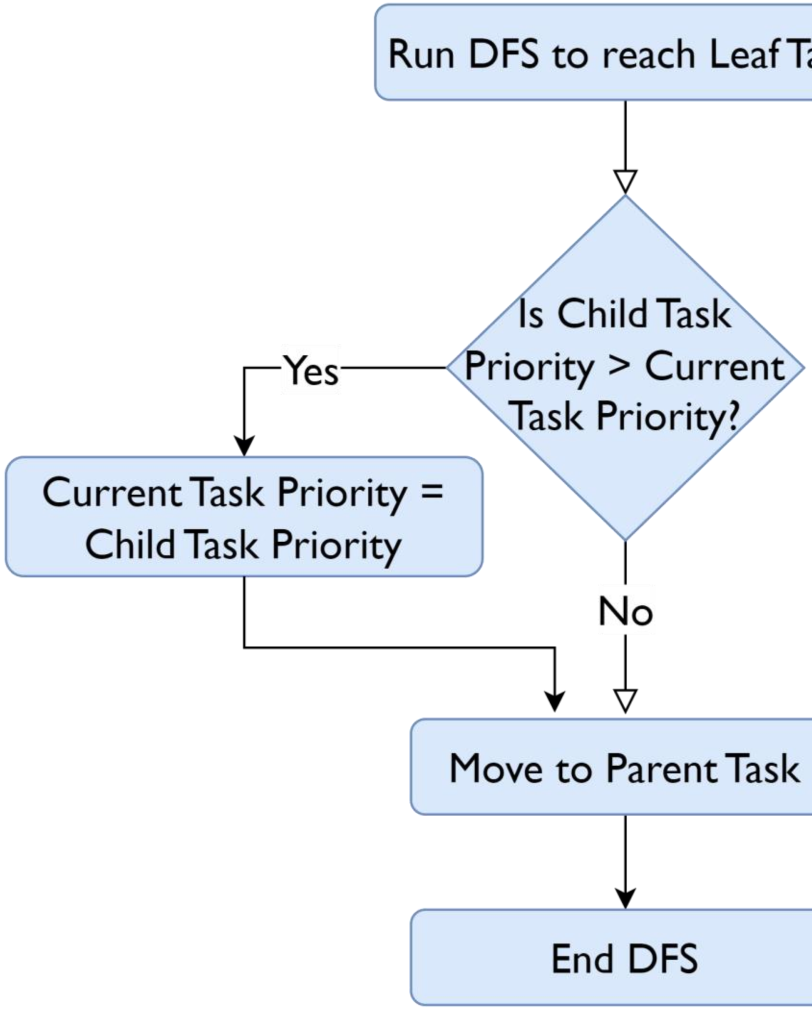


Fig: Task Priority Update Flowchart

UPDATE TASK PRIORITY

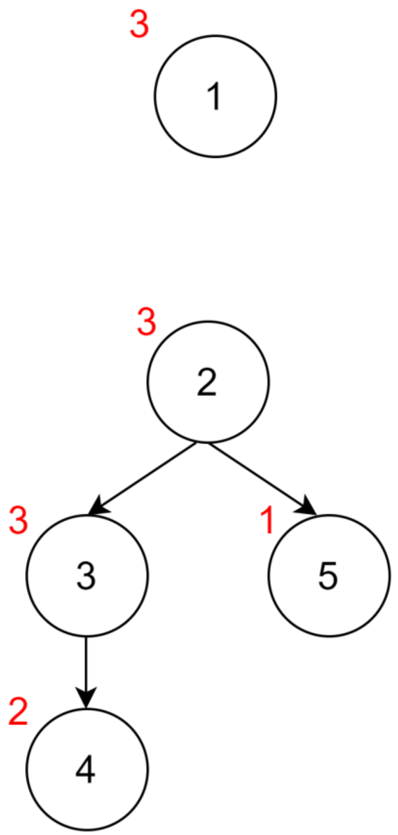


Fig: DAG Example

Task ID	Original Priority
1	3
2	3
3	3
4	2
5	1

UPDATE TASK PRIORITY

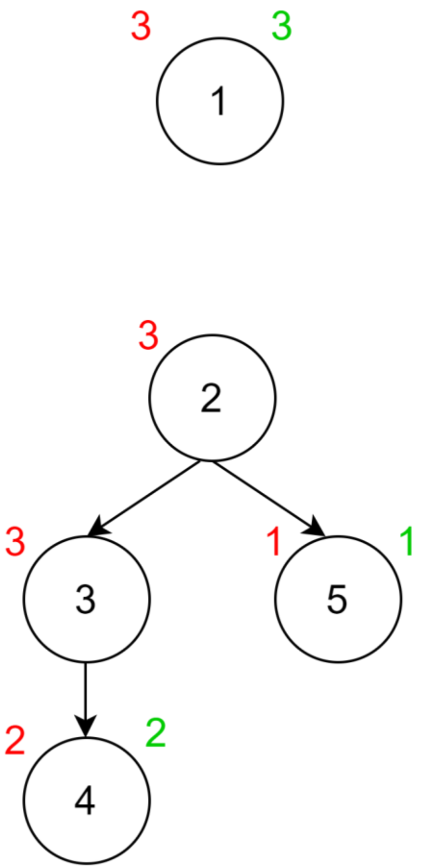


Fig: DAG Example

Task ID	Original Priority	Imposed Priority
1	3	3
2	3	
3	3	
4	2	2
5	1	1

UPDATE TASK PRIORITY

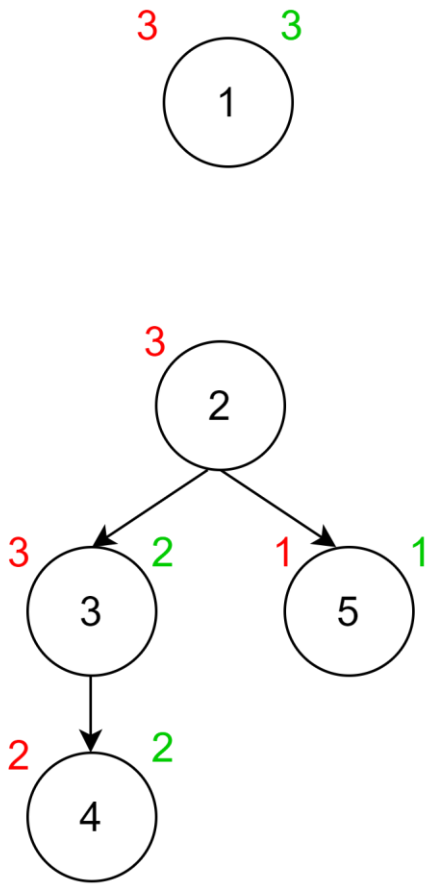


Fig: DAG Example

Task ID	Original Priority	Imposed Priority
1	3	3
2	3	1
3	3	2
4	2	2
5	1	1

UPDATE TASK PRIORITY

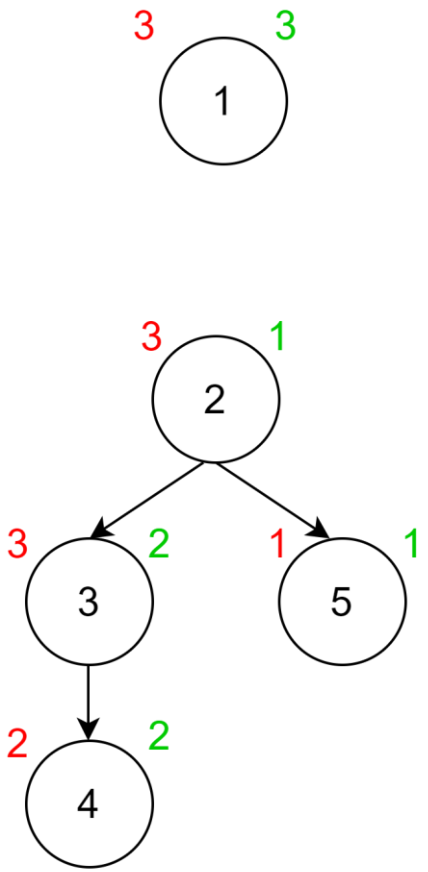


Fig: DAG Example

Task ID	Original Priority	Imposed Priority
1	3	3
2	3	1
3	3	2
4	2	2
5	1	1

PRIORITY-DEPENDENCY SCORE (PD_SCORE) CALCULATE

Set PD_Score = 0

PD_Score += ChildTask.PD_Score() + (4-ChildTask.Priority())

Current Task PD_Score = PD_Score

A fog node receives a batch of tasks

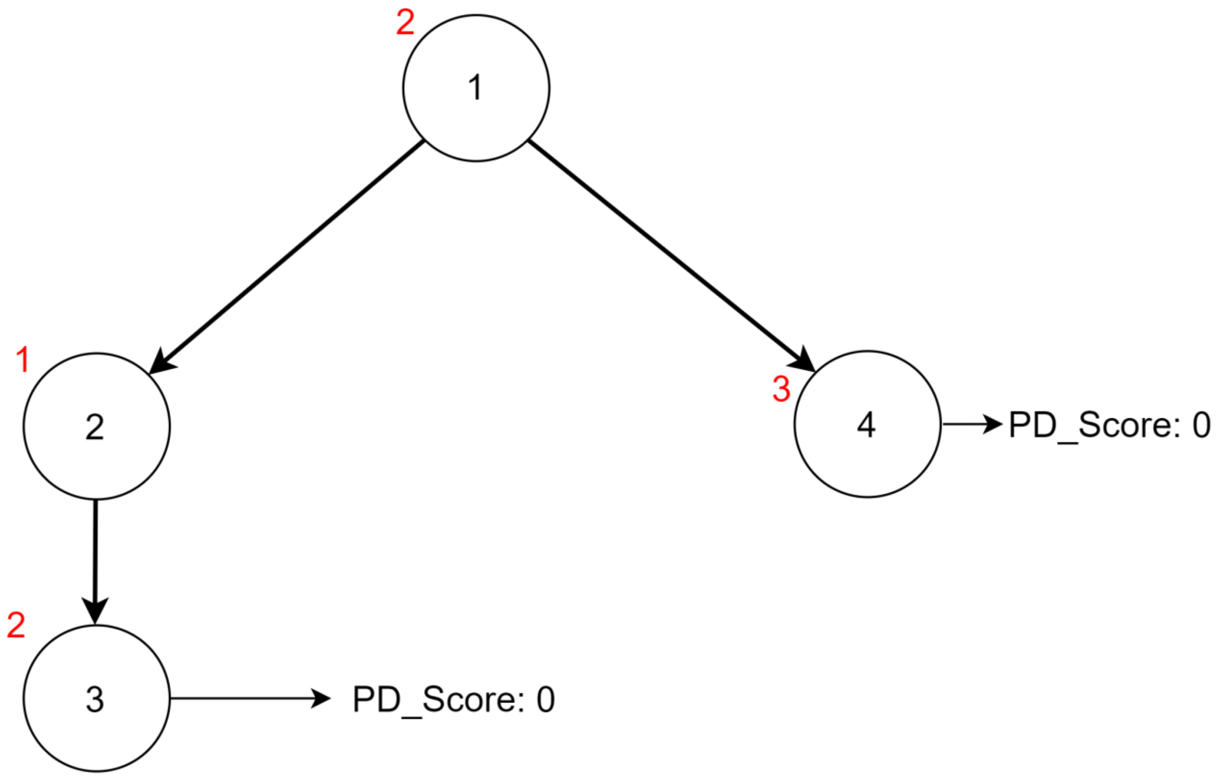
Task priority is updated if needed

Priority-Dependency score is calculated for every task

Tasks are inserted into respective high, mid and low priority level queues (waiting and ready)

Task is selected from ready queue based on deadline and PD_score and sent to the respective VM for execution

PRIORITY-DEPENDENCY SCORE (PD_SCORE) CALCULATE



Set PD_Score = 0

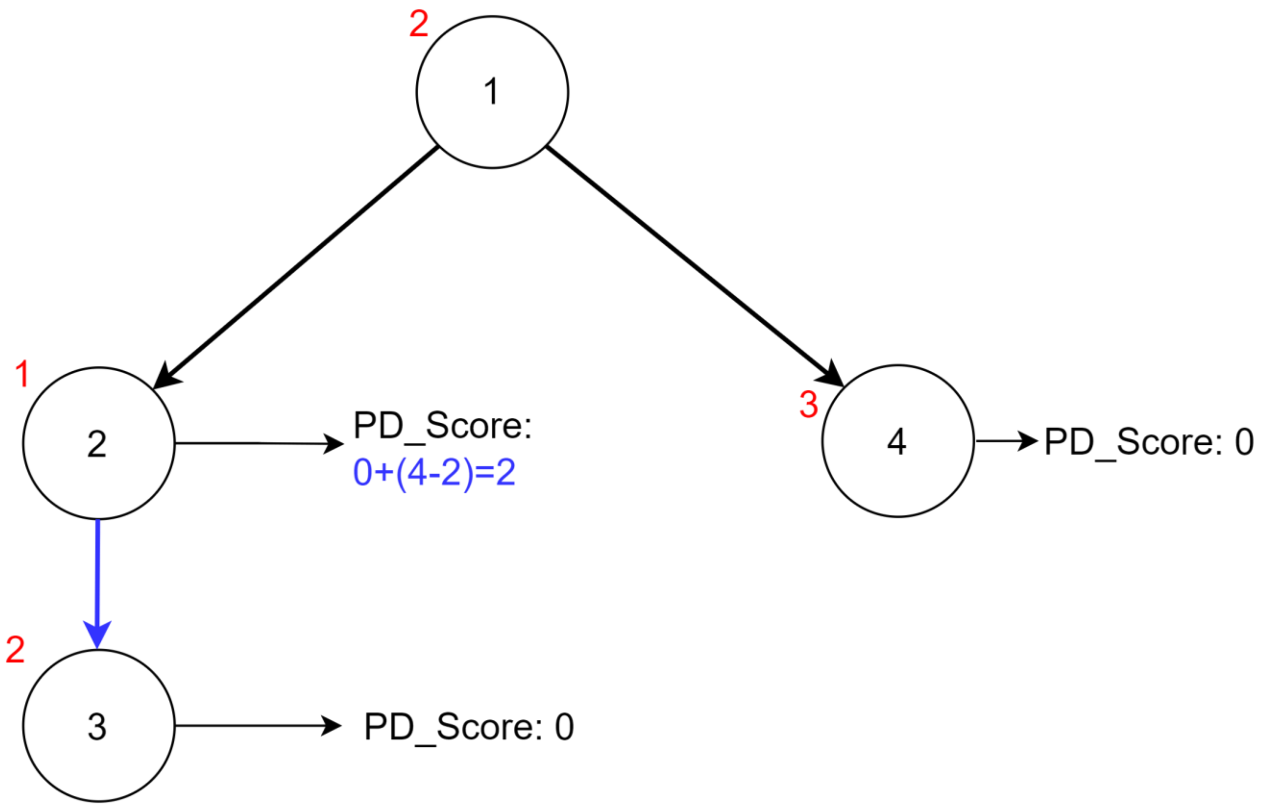
↓

PD_Score += ChildTask.PD_Score() + (4 - ChildTask.Priority())

↓

Current Task PD_Score = PD_Score

PRIORITY-DEPENDENCY SCORE (PD_SCORE) CALCULATE

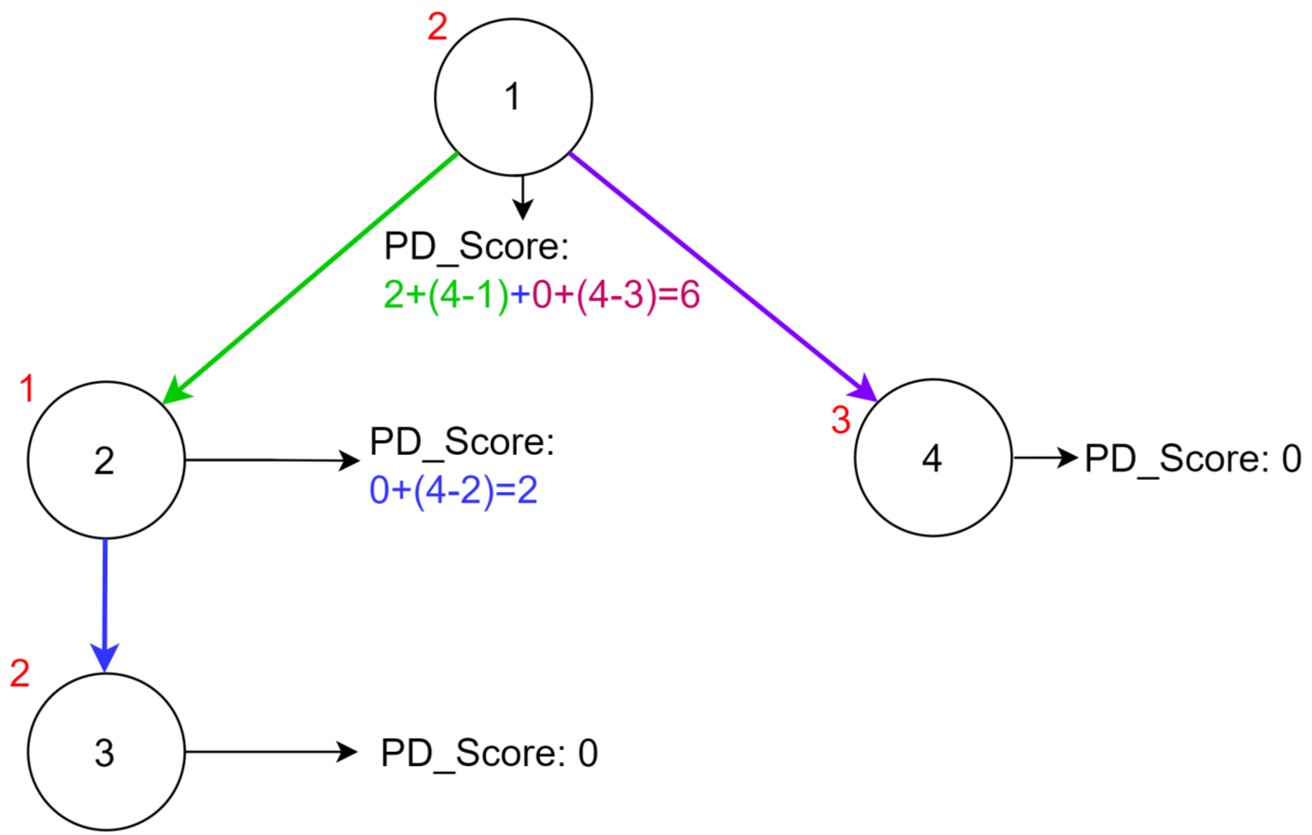


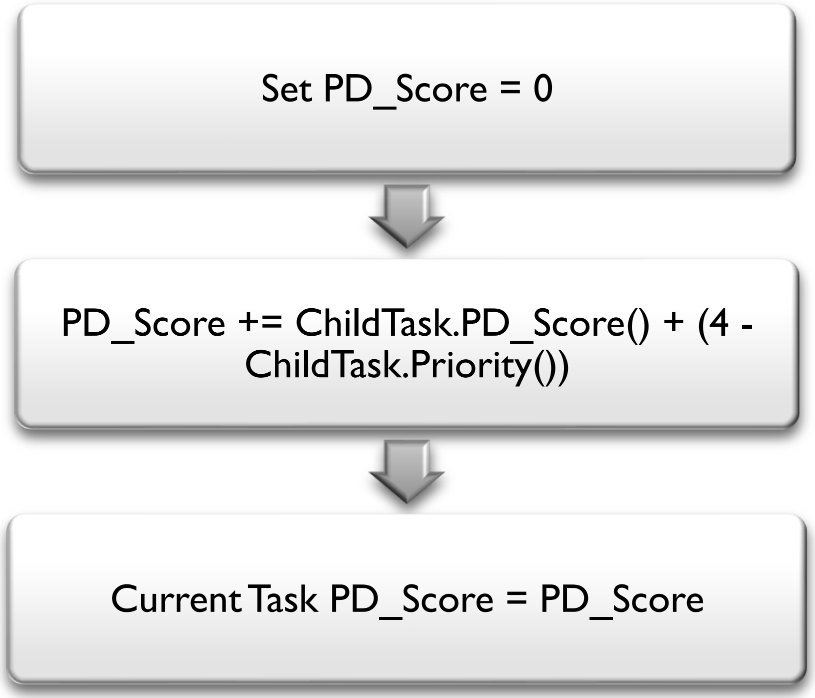
Set PD_Score = 0

PD_Score += ChildTask.PD_Score() + (4 - ChildTask.Priority())

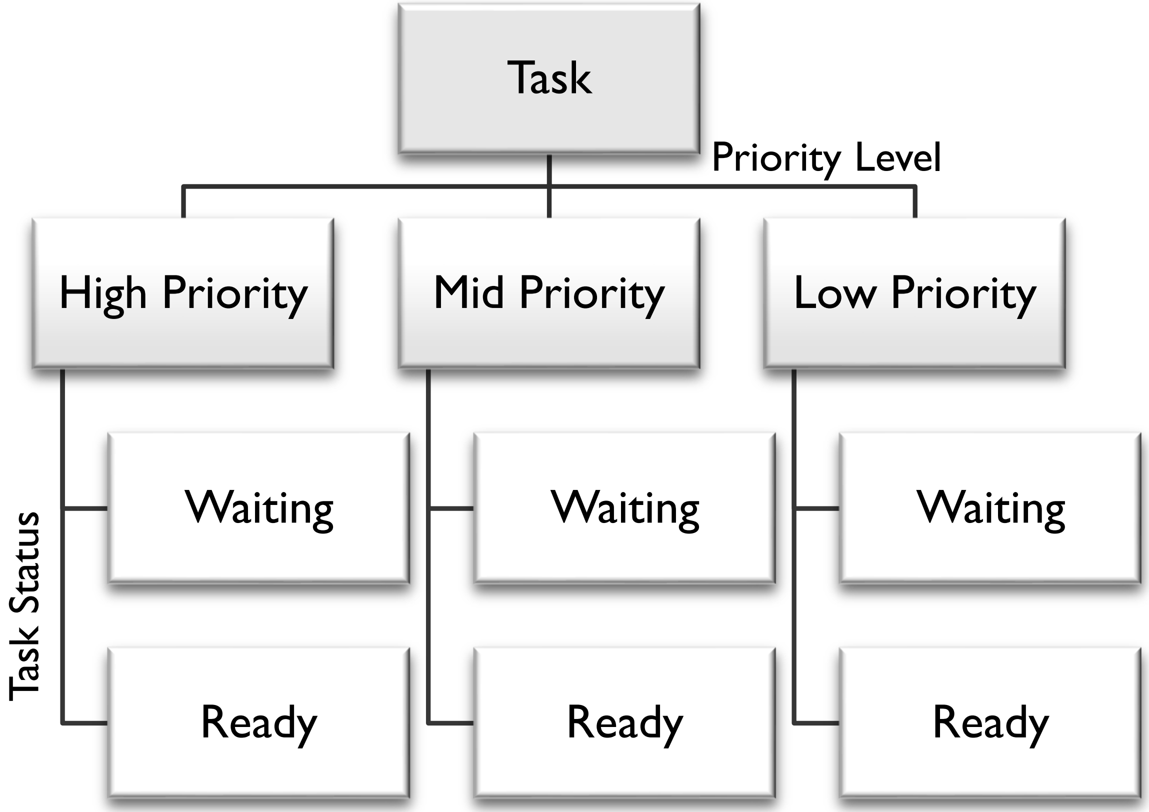
Current Task PD_Score = PD_Score

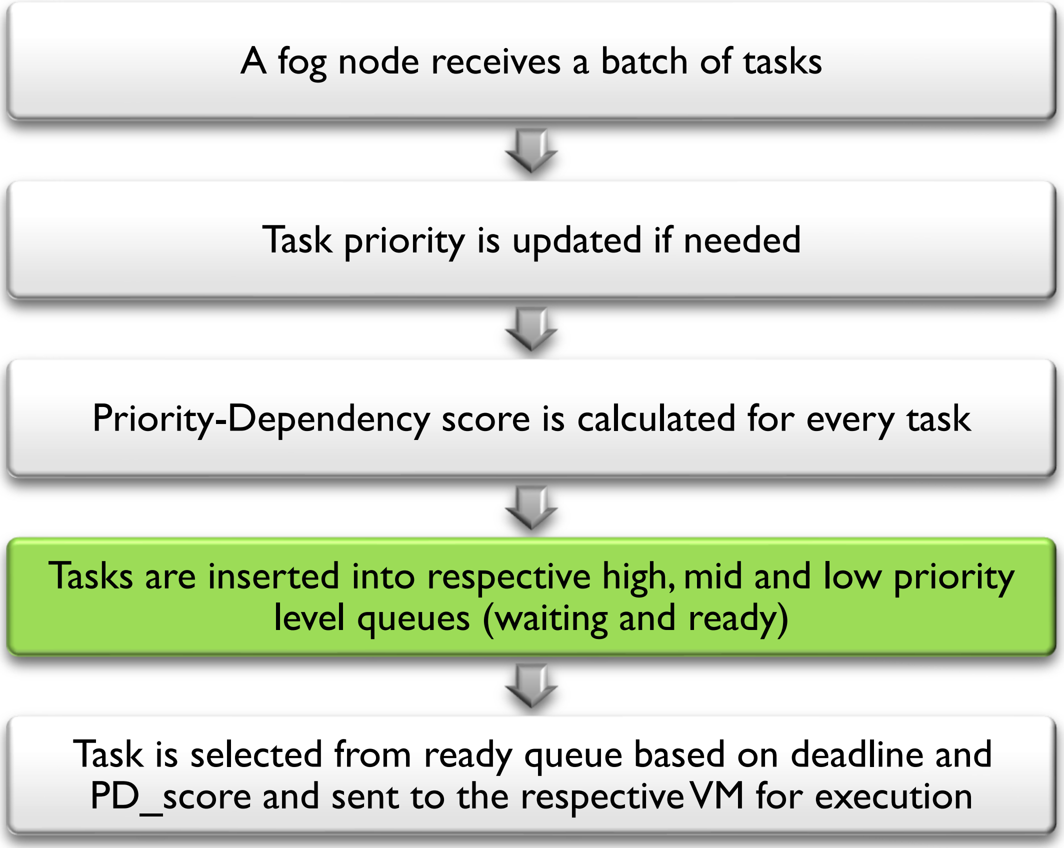
PRIORITY-DEPENDENCY SCORE (PD_SCORE) CALCULATE



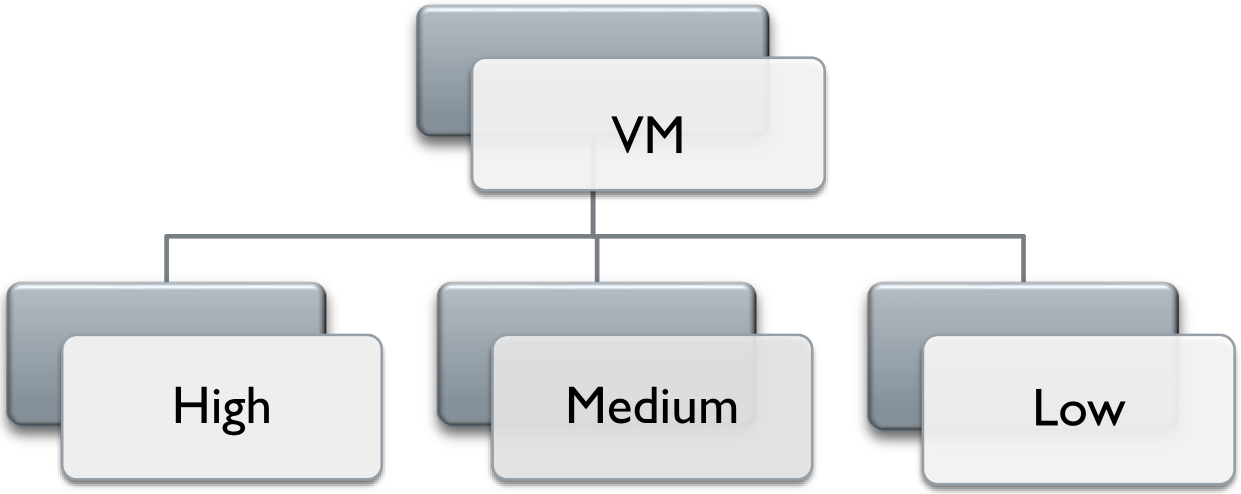


TASKS CATEGORIES





VM CATEGORIES



A fog node receives a batch of tasks

↓

Task priority is updated if needed

↓

Priority-Dependency score is calculated for every task

↓

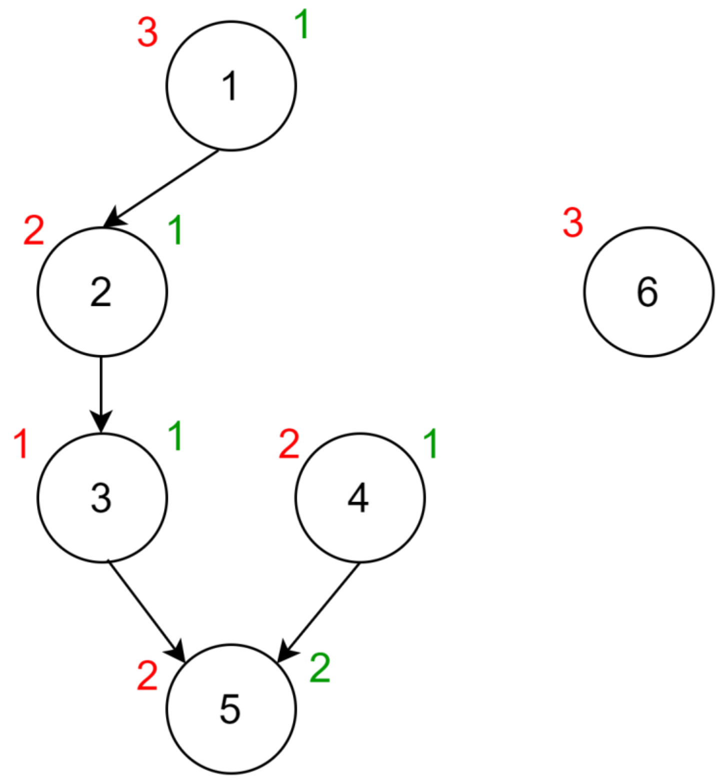
Tasks are inserted into respective high, mid and low priority level queues (waiting and ready)

↓

Task is selected from ready queue based on deadline and PD_score and sent to the respective VM for execution

SAMPLE TASKSET

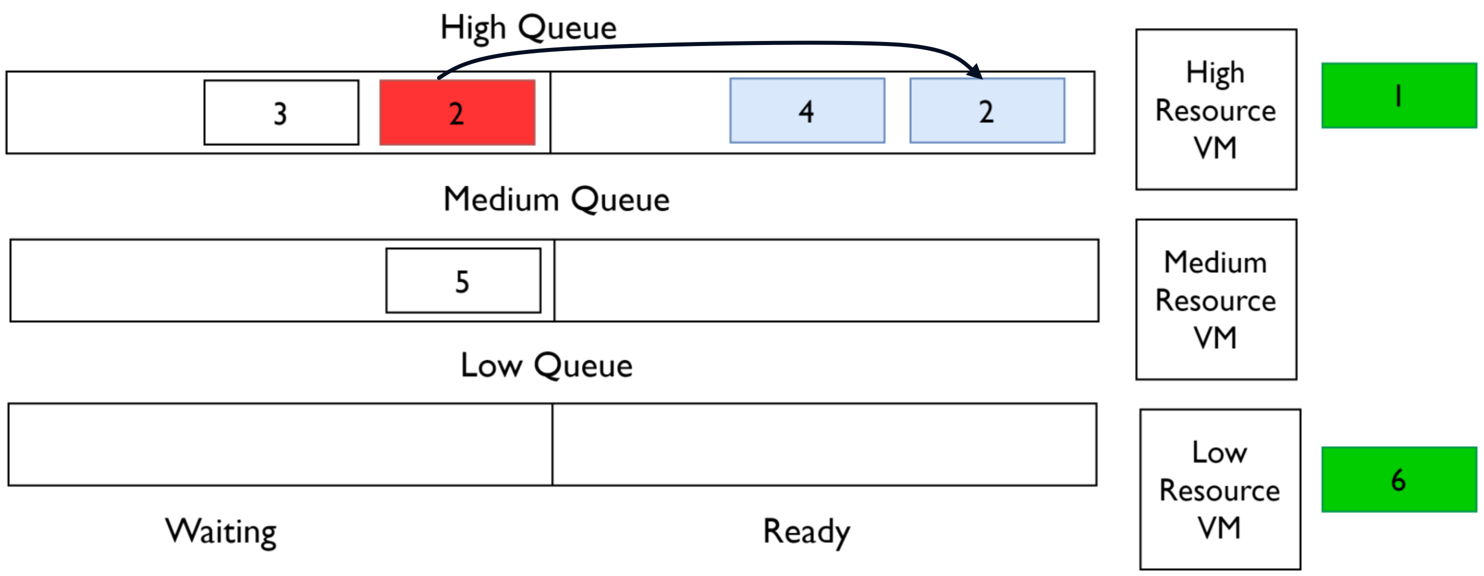
Task ID	Priority Level	Length (MI)	Deadline	Predecessor List	Imposed Priority	PD_Score
1	3	8000	10		1	7
2	2	10000	12	1	1	5
3	1	24000	26	2	1	2
4	1	9000	12		1	2
5	2	12000	30	3, 4	2	0
6	3	10000	40		3	0



TASK EXECUTION

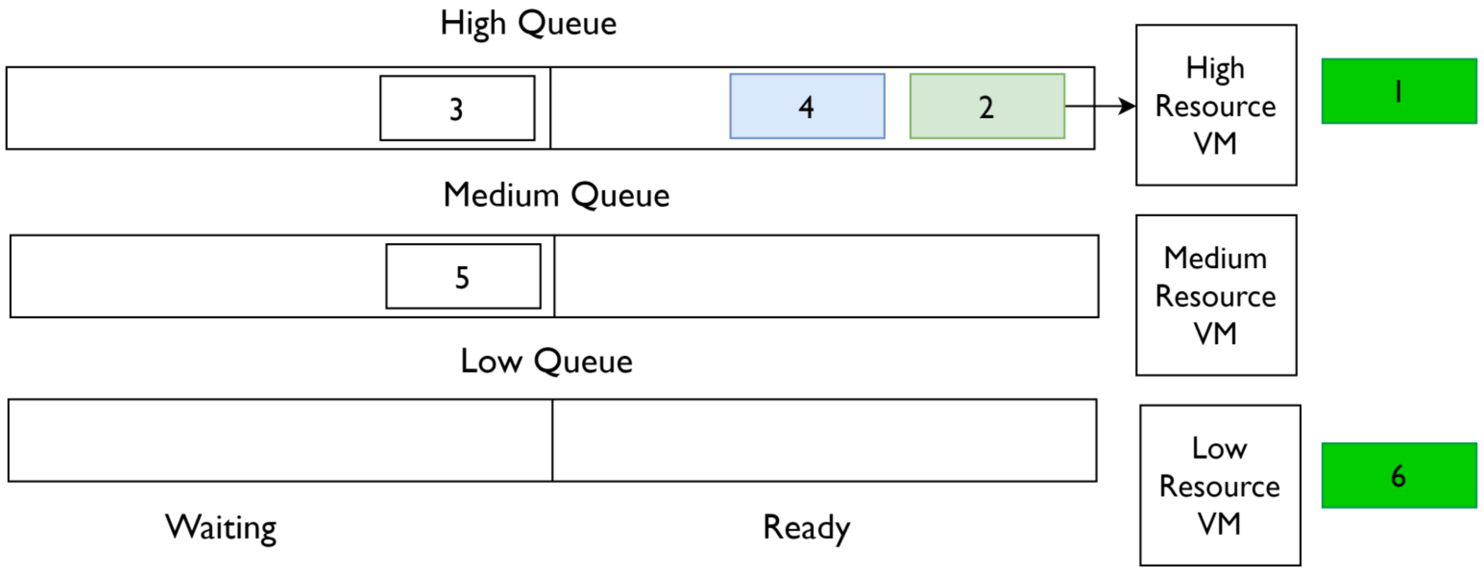
Task ID	Deadline	Predecessor List	Imposed Priority	PD_Score
1	10		1	7
2	12	1	1	5
3	26	2	1	2
4	12		1	2
5	30	3, 4	2	0
6	40		3	0

TASK EXECUTION



Task ID	Deadline	Predecessor List	Imposed Priority	PD_Score
1	10		1	7
2	12	1	1	5
3	26	2	1	2
4	12		1	2
5	30	3, 4	2	0
6	40		3	0

TASK EXECUTION



Task ID	Deadline	Predecessor List	Imposed Priority	PD_Score
1	10		1	7
2	12	1	1	5
3	26	2	1	2
4	12		1	2
5	30	3, 4	2	0
6	40		3	0

EXPERIMENTS AND RESULTS

EXPERIMENTAL SETUP

Evaluation Metrices

- $Task\ Completion\ Rate = \dfrac{Tasks\ Completed\ on\ Time}{Total\ Number\ of\ Tasks}$
- $Response\ Time = \dfrac{Start\ Time\ - Arrival\ Time}{Total\ Number\ of\ Tasks}$
- $Makespan = Max(End\ Time) - Min(Start\ Time)$
- $Throughput = \dfrac{Time\ of\ Tasks\ Completed\ on\ Time}{Total\ Time\ to\ Complete\ all\ Tasks}$

Hyper-parameters

- *Total Number of Tasks*
- *Dependent Task Ratio*
- *Number of Max Parent of a Task*
- *Task Priority Level Ratio*

ALGORITHMS

- Proposed: Scoring with Heterogeneous Multi VMs
- Prioritized Task Scheduling without scoring (Choudhari [2])
- Scoring with Homogeneous Multi VMs
- Scoring with Single VM

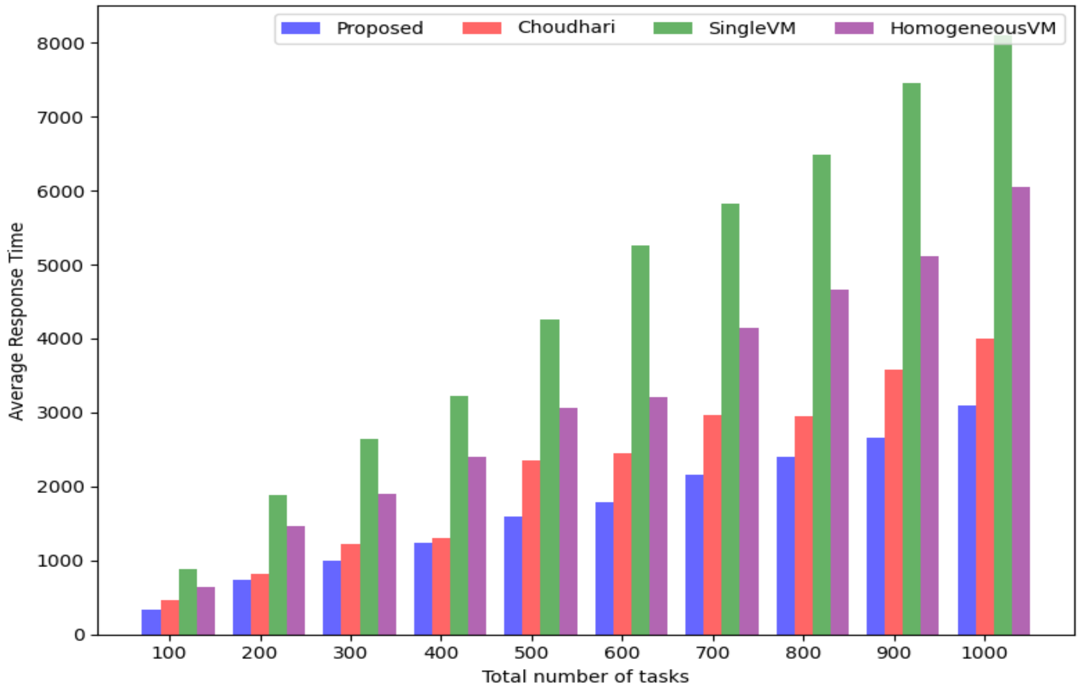
EXPERIMENT - I

Hyperparameters:

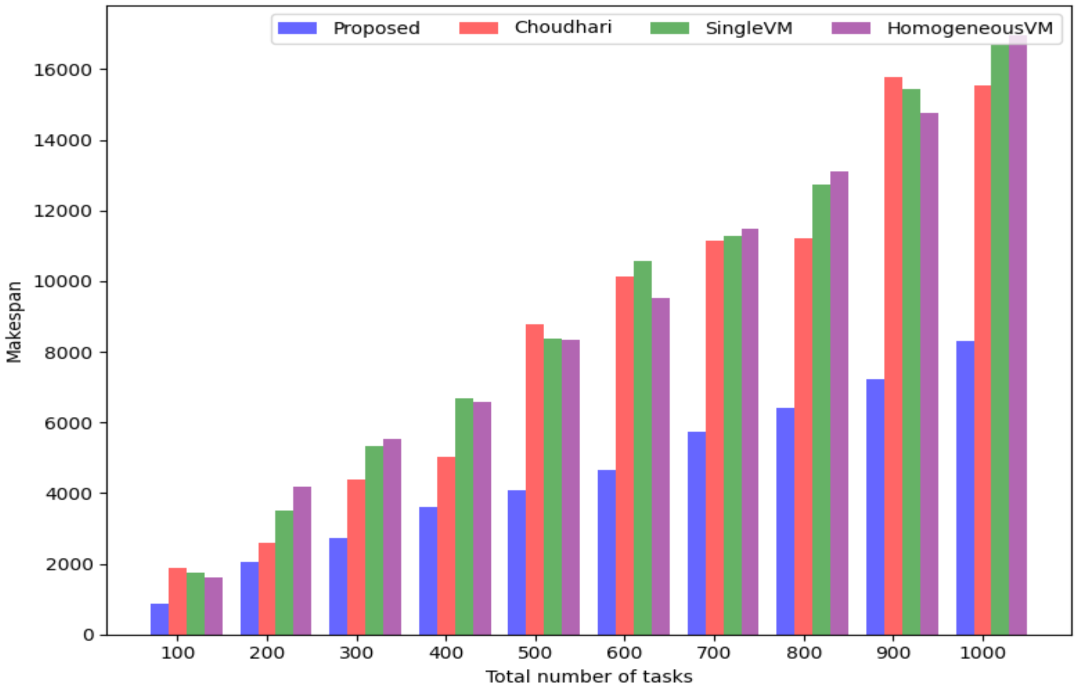
- Fixed
  - Dependent Task Ratio = 50%
  - Priority Level Ratio $[w_H = 0.4, w_M = 0.35, w_L = 0.25]$
  - Number of Predecessor for a Task $(min = 1, max = 3)$
- Variable:
  - Number of Tasks : $100 - 1000$

EXP-I: VARYING TOTAL NUMBER OF TASKS

Parameters: Dependent Task Ratio = 50% Task Priority Ratio = [0.4, 0.35, 0.25] Num_predecessors = {1, 3}



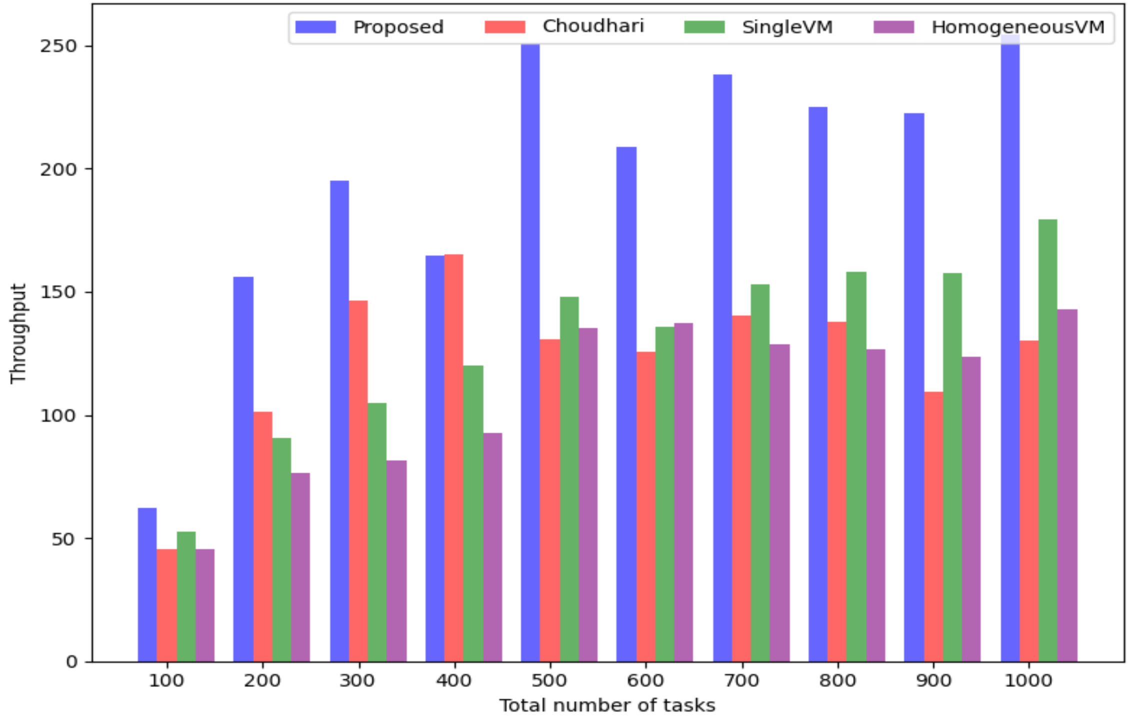
Average Response Time vs Total Number of Tasks



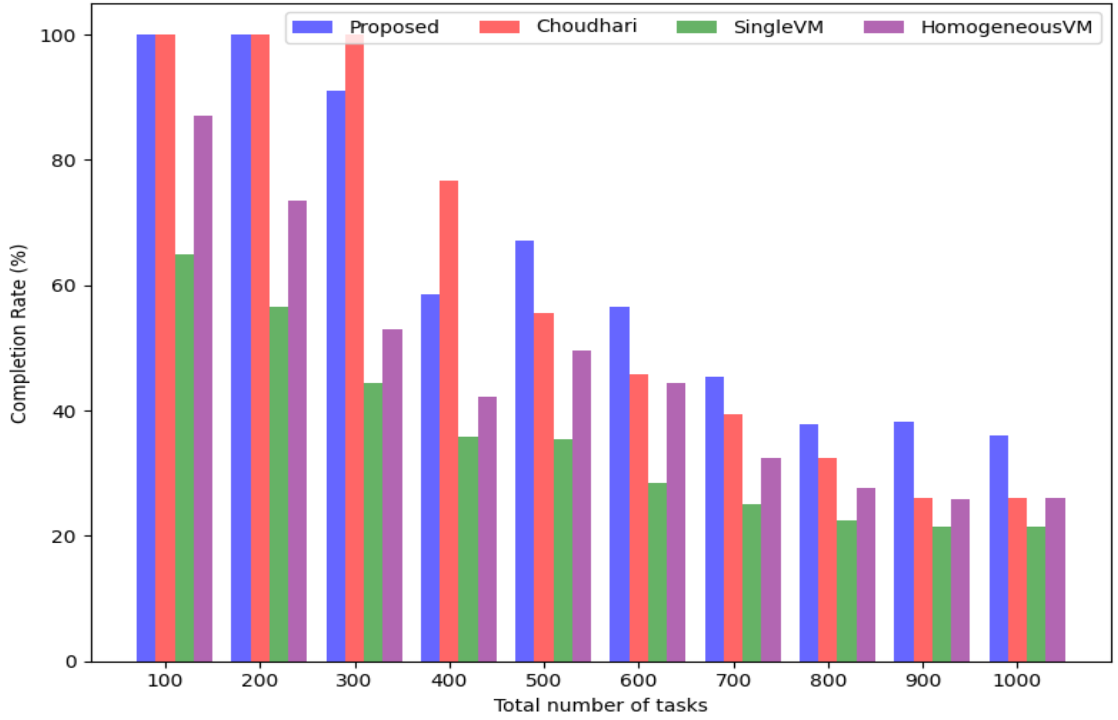
Makespan vs Total Number of Tasks

EXP-I: VARYING TOTAL NUMBER OF TASKS

Parameters: Dependent Task Ratio = 50% Task Priority Ratio = [0.4, 0.35, 0.25] Num_predecessors = {1, 3}



Throughput vs Total Number of Tasks



Task Completion Rate vs Total Number of Tasks

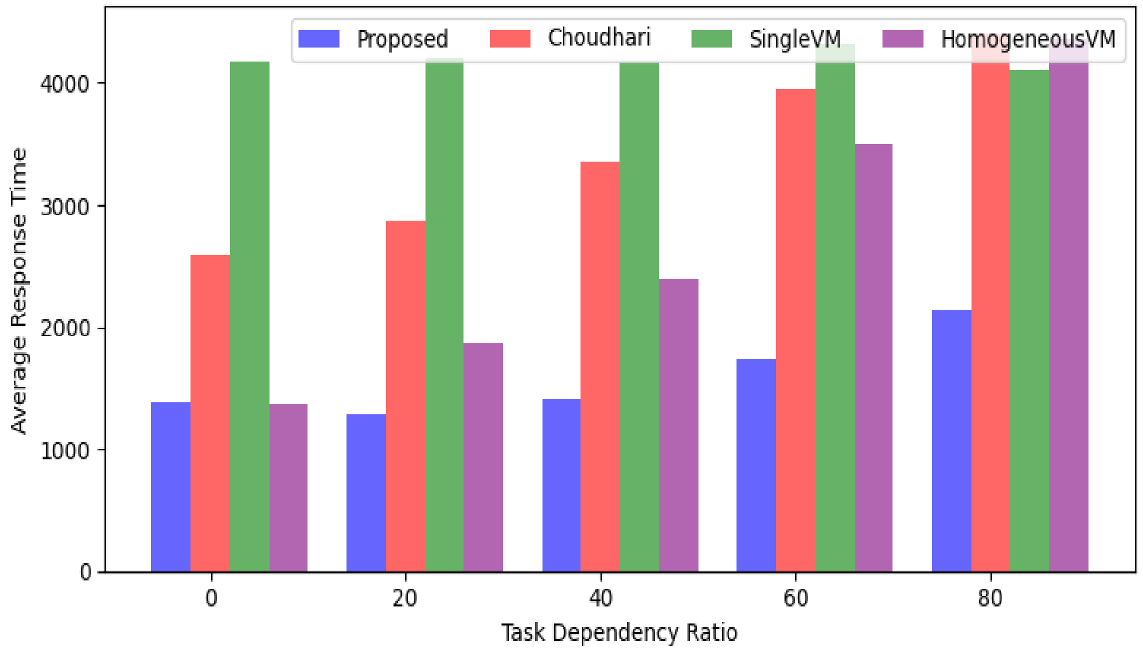
EXPERIMENT - II

Hyperparameters:

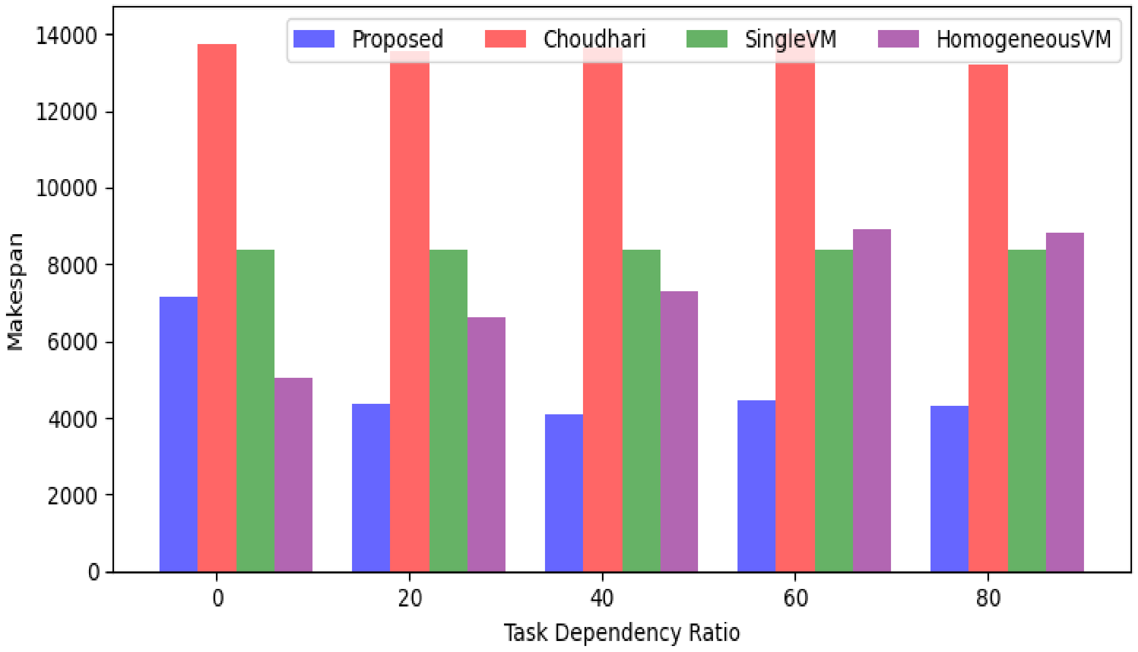
- Fixed
  - Number of Tasks = 500
  - Priority Level Ratio $[w_H = 0.4, w_M = 0.35, w_L = 0.25]$
  - Number of Predecessor for a Task $(min = 1, max = 3)$
- Variable:
  - Dependent Task Ratio: 0% - 80%

EXP-II: VARYING DEPENDENT TASK RATIO

Parameters: Number of Tasks = 500 Task Priority Ratio = [0.4, 0.35, 0.25] Num_predecessors = {1, 3}



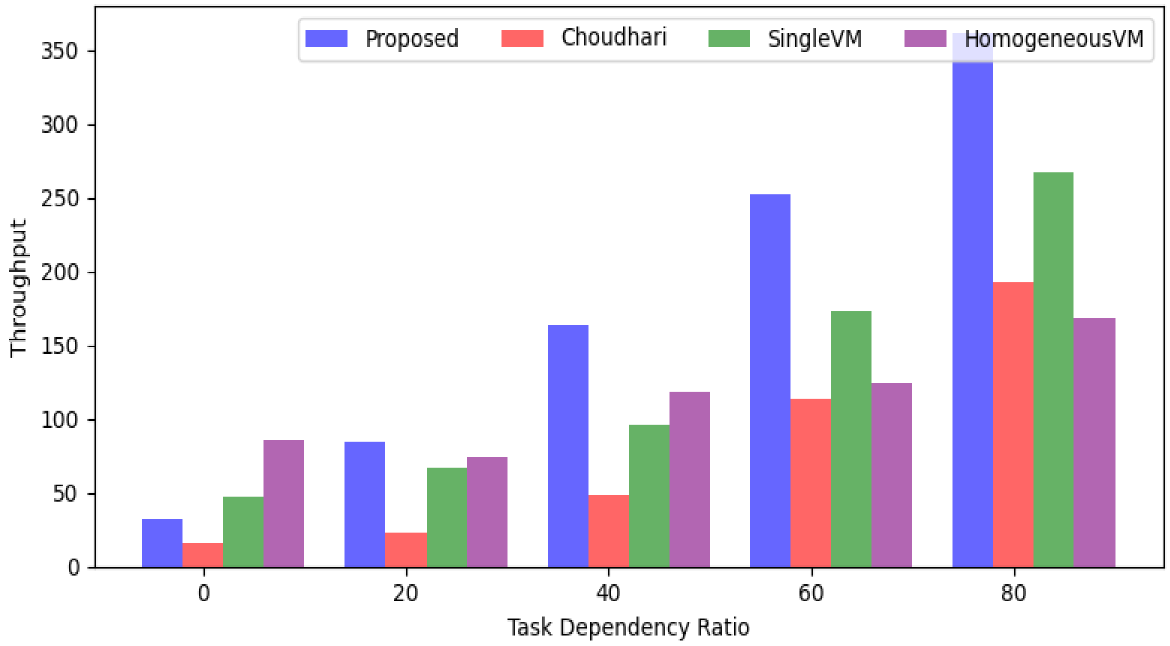
Average Response Time vs Dependent Task Ratio



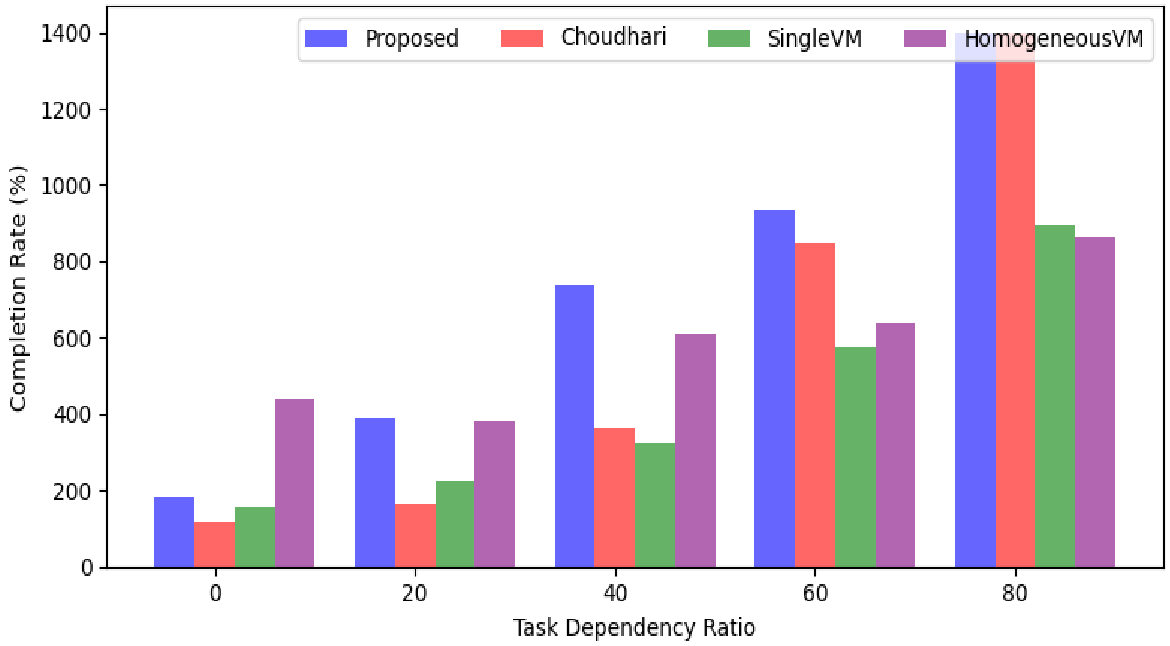
Makespan vs Dependent Task Ratio

EXP-II: VARYING DEPENDENT TASK RATIO

Parameters: Number of Tasks = 500 Task Priority Ratio = [0.4, 0.35, 0.25] Num_predecessors = {1, 3}



Throughput vs Dependent Task Ratio



Task Completion Rate vs Dependent Task Ratio

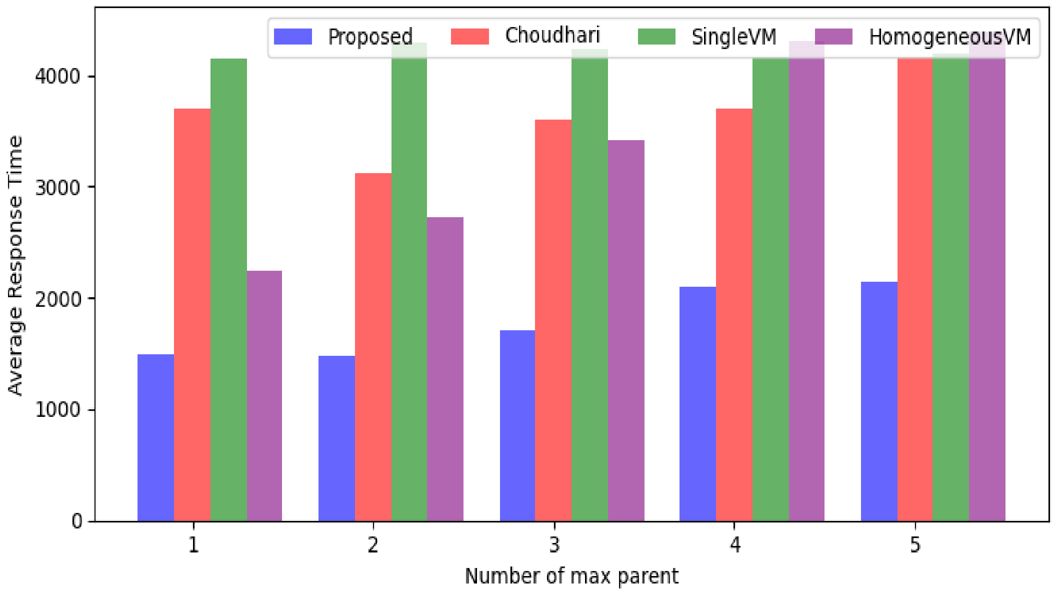
EXPERIMENT - III

Hyperparameters:

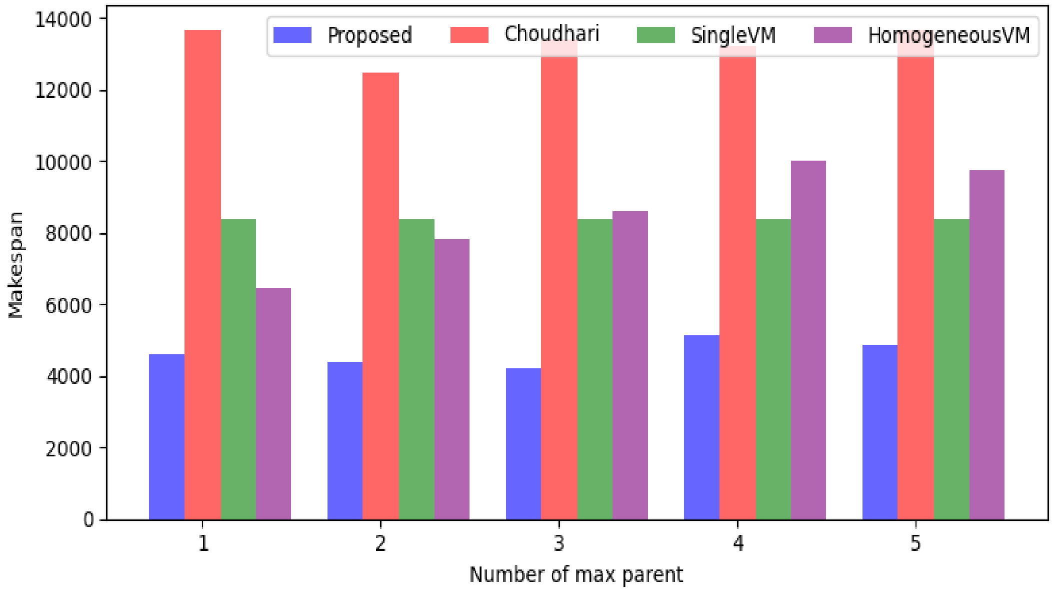
- ❑ Fixed
  - Number of Tasks = 500
  - Dependent Task Ratio = 60%
  - Priority Level Ratio $[w_H = 0.4, w_M = 0.35, w_L = 0.25]$
- ❑ Variable:
  - Number of Predecessor for a Task $(min = 1, max = 1 - 5)$

EXP-III: VARYING MAX NUMBER OF PREDECESSOR OF A TASK

Parameters: Number of Tasks = 500 Dependent Task Ratio = 60% Task Priority Ratio = [0.4, 0.35, 0.25]



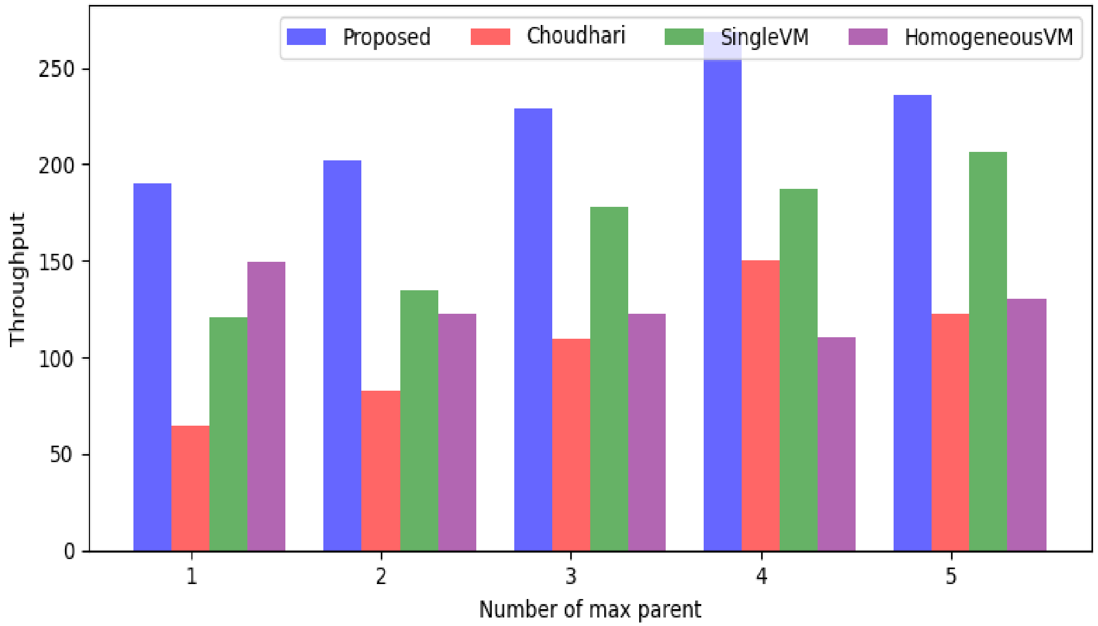
Average Response Time vs Number of Max Predecessor



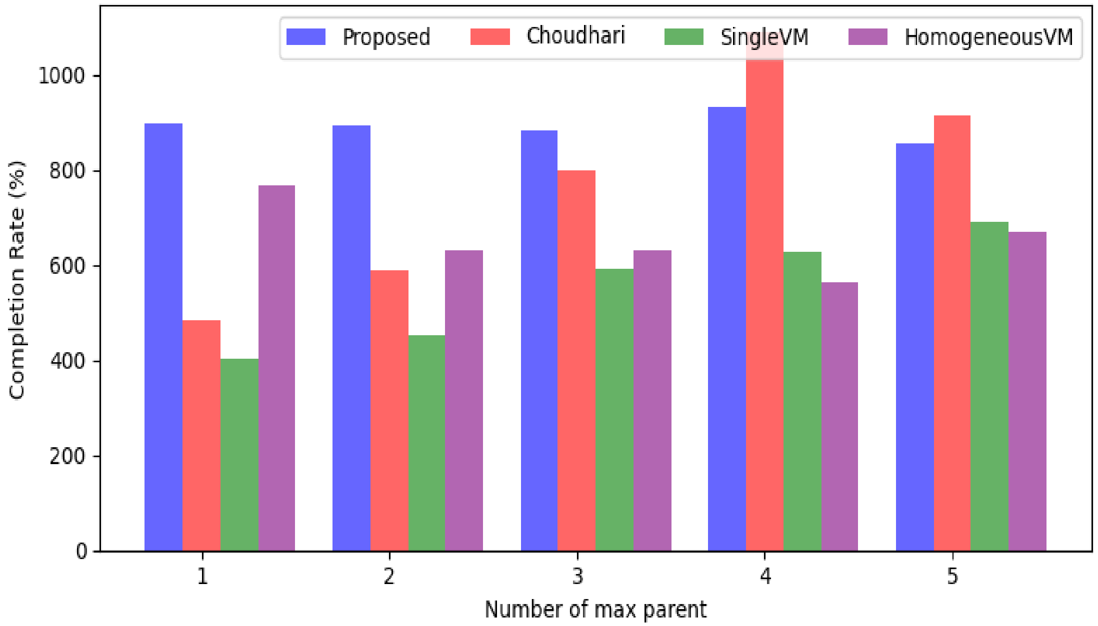
Makespan vs Number of Max Predecessor

EXP-III: VARYING MAX NUMBER OF PREDECESSOR OF A TASK

Parameters: Number of Tasks = 500 Dependent Task Ratio = 60% Task Priority Ratio = [0.4, 0.35, 0.25]



Throughput vs Number of Max Predecessor



Task Completion Rate vs Number of Max Predecessor

EXPERIMENT - 4

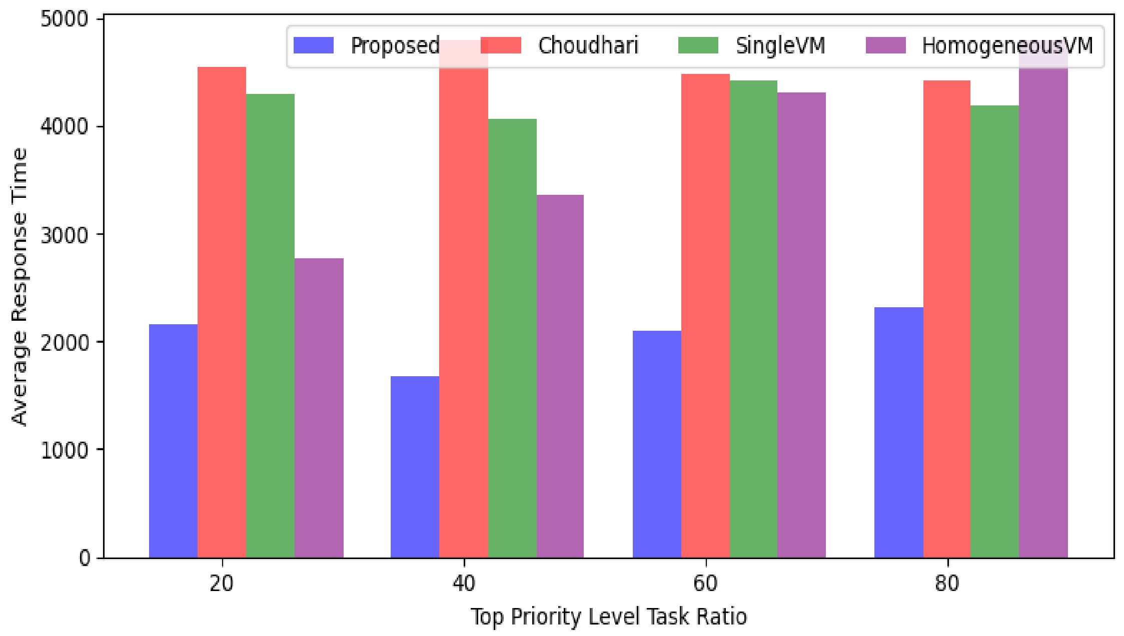
Hyperparameters:

- ❑ Fixed
  - Number of Tasks = 500
  - Dependent Task Ratio = 60%
  - Number of Predecessor for a Task $(min = 1, max = 3)$
- ❑ Variable:
  - Priority Level Ratio

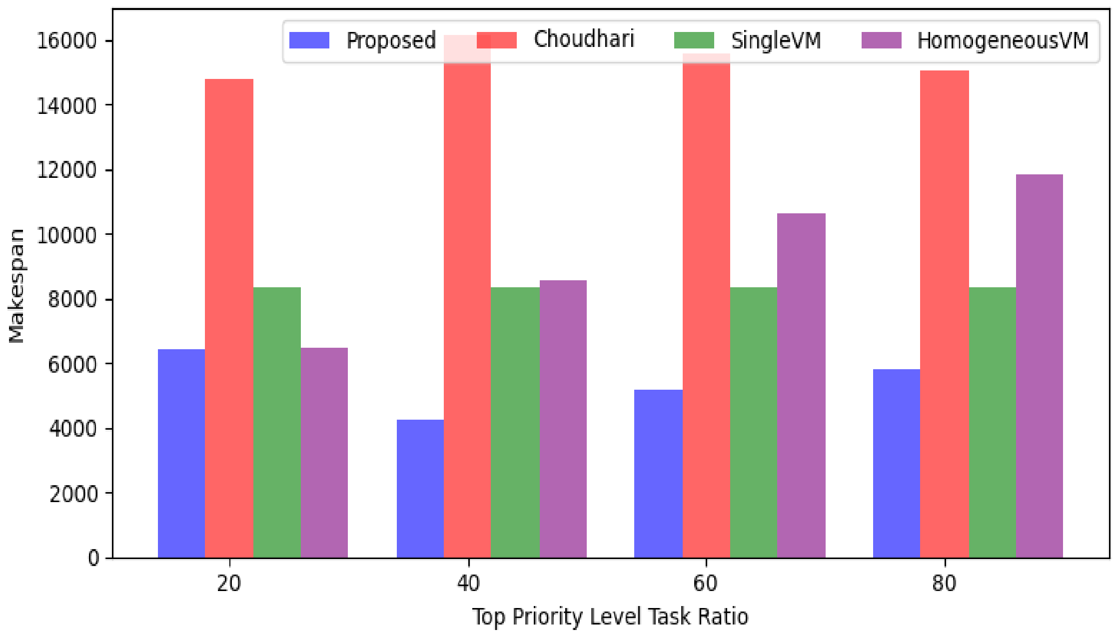
$$[w_H = 0.2 - 0.8, w_M = 0.8 - 0.2, w_L = 1 - w_H - w_M]$$

EXP-IV: VARYING TASK PRIORITY RATIO

Parameters: Number of Tasks = 500 Dependent Task Ratio = 60% Num_predecessors = {1, 3}



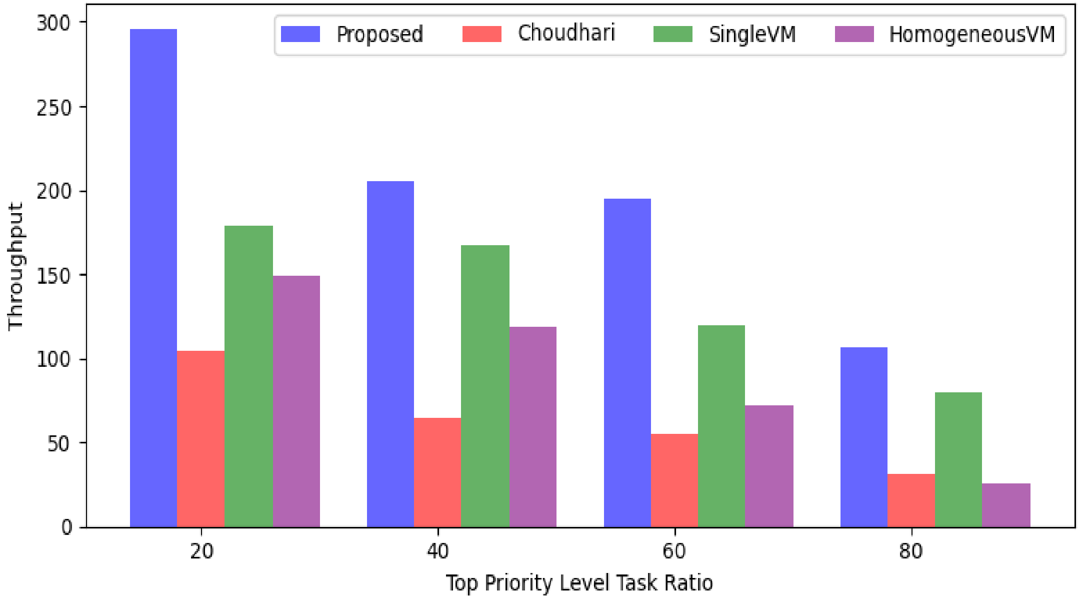
Average Response Time vs Task Priority Ratio



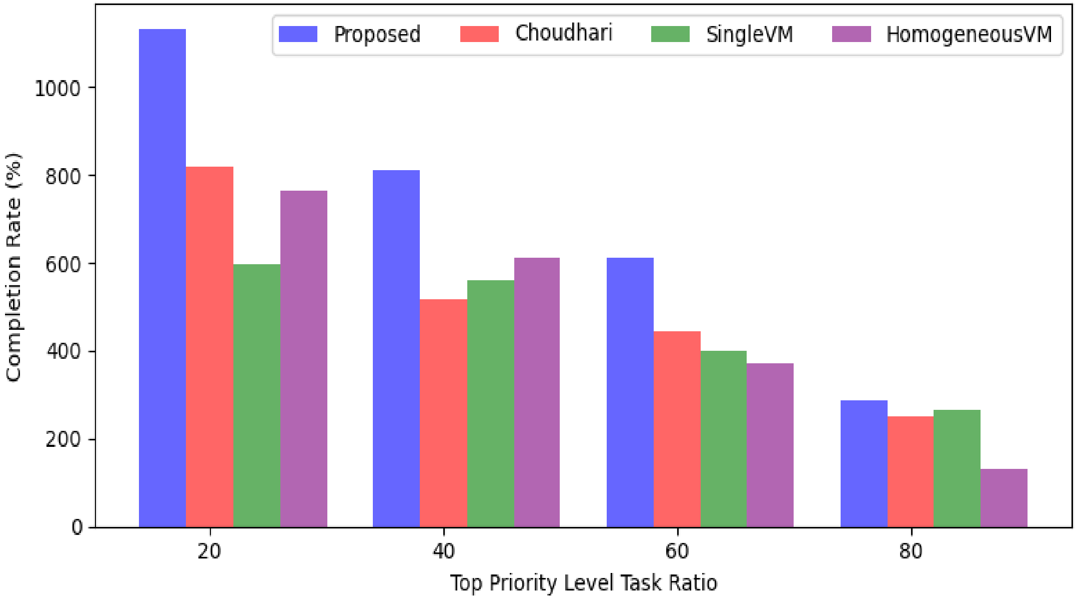
Makespan vs Task Priority Ratio

EXP-IV: VARYING TASK PRIORITY RATIO

Parameters: Number of Tasks = 500 Dependent Task Ratio = 60% Num_predecessors = {1, 3}



Throughput vs Task Priority Ratio



Task Completion Rate vs Task Priority Ratio

EXPERIMENT SUMMARY

Metric / Hyper Parameters	Response Time	Makespan	Throughput	Task Completion Rate
Number of tasks (100-1000)	Proposed	Proposed	Proposed	(100-200) Proposed & Choudhari (300-400) Choudhari (500-1000) Proposed
Dependent task ratio (0%-80%)	0% Homogeneous, (20-80)% Proposed	0% Homogeneous, (20-80)% Proposed	0% Homogeneous, (20-80)% Proposed	0% Homogeneous, (20-80)% Proposed
max parent of a task (1-5)	Proposed	Proposed	Proposed	(1-3) Proposed (4-5) Choudhari
High priority tasks ratio (20%-80%)	Proposed	Proposed	Proposed	Proposed

FUTURE WORKS

- Fog Resource Management
- Task Preemption
- Fault Tolerance

REFERENCES (RELATED WORKS)

1. Elshahed, E. M., Abdelmoneem, R. M., Shaaban, E., Elzahed, H. A., & Al-Tabbakh, S. M. (2023). Prioritized scheduling technique for healthcare tasks in cloud computing. The Journal of Supercomputing, 79(5), 4895-4916.
2. Choudhari, T., Moh, M., & Moh, T. S. (2018, March). Prioritized task scheduling in fog computing. In Proceedings of the ACMSE 2018 Conference (pp. 1-8).
3. Chronaki, K., Rico, A., Badia, R. M., Ayguadé, E., Labarta, J., & Valero, M. (2015, June). Criticality-aware dynamic task scheduling for heterogeneous architectures. In Proceedings of the 29th ACM on International Conference on Supercomputing (pp. 329-338).
4. Fellir, F., El Attar, A., Nafil, K., & Chung, L. (2020, February). A multi-Agent based model for task scheduling in cloud-fog computing platform. In 2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT) (pp. 377-382). IEEE.

REFERENCES (MISC)

- Hameed, K., Bajwa, I. S., Ramzan, S., Anwar, W., & Khan, A. (2020). An intelligent IoT based healthcare system using fuzzy neural networks. Scientific programming, 2020(1), 8836927.
- T. Michailidis, E., G. Kogias, D., & Voyiatzis, I. (2020, November). A review on hardware security countermeasures for iot: Emerging mechanisms and machine learning solutions. In Proceedings of the 24th Pan-Hellenic Conference on Informatics (pp. 268-271).
- Masri, W., Al Ridhawi, I., Mostafa, N., & Pourghomi, P. (2017, July). Minimizing delay in IoT systems through collaborative fog-to-fog (F2F) communication. In 2017 ninth international conference on ubiquitous and future networks (ICUFN) (pp. 1005-1010). IEEE.
- https://medium.com/@sahilmjain03/cloud-computing-and-iot-2edd5080a7ea