CS_FedSIM's documentation!

Contents

CS_FedSIM is a simulator framework of client selection in FL that handles resource consumption.

class `node.`**Node**(*name: str, data: Optional[str] = None, data_type: Optional[str] = None, mobility_mode: bool = False*)

Bases: `object`

A class that represents the basis of the node's module.

…

name

The name of the node to be identified with.

Type: str

data

The data available in the node.

Type: str

data_type

The type of the data in the node.

Type: str

mobility_mode

Indicates if the node is mobile or not.

Type: bool

node_id

The node identifier, represented by a hash.

Type: str

ip_addr

The IP address of the node in the network.

Type: str

status

Indicates the status of the node if it is available and can be contacted. (1 = Yes, 0 = No)

Type: int

`leader`

Indicates whether the node is the leader of its network or not.

Type: bool

`gathered_data`

Data collected from other nodes or the server.

Type: any

`battery_usage`

The percentage of the node's battery usage.

Type: int

`total_energy`

The total energy capacity of the node.

Type: float

`energy_consumption`

Indicates how much energy this node consumes in an operation.

Type: float

`current_energy`

The current energy capacity in mAh.

Type: float

`total_storage`

The total storage capacity of the node.

Type: int

`current_storage`

The current storage capacity of the node.

Type: int

`cpu_power`

The total power of the processor.

Type: float

`cpu_usage`

The percentage of CPU usage.

Type: int

`memory`

The memory capacity of the node.

Type: int

`memory_usage`

The memory usage percentage of the node.

Type: int

`get_resources_information():`

Take all the information about the resources of this node.

Notes

The other methods are a group of getters and setters, so they are not explained for this module.

`__init__(name: str, data: Optional[str] = None, data_type: Optional[str] = None, mobility_mode: bool = False)`

Constructs all the necessary attributes for the node object.

Parameters:
- **name** (*str*) – The name of the node to be identified with.
- **data** (*str, optional*) – The data available in the node.
- **data_type** (*str, optional*) – The type of the data in the node.
- **mobility_mode** (*bool, optional*) – Indicates if the node is mobile or not.

Examples

```
>>> node = Node(name='node1')
```

`get_resources_information()`

Take all the information about the resources of this node.

Returns:	**information (tuple)**
Return type:	All resource information represented by a tuple of 13 information.

Examples

```
>>> node.get_resources_information()
```

class `node.PowNode(name: str, mobility_mode: bool = False)`

Bases: `node.Node.Node`

A class that represents the module of a powerful node.

…

`name`

The node of the node to identify it.

Type: str

`mobility_mode`

Indicates whether the node is mobile or stationary.

Type: bool, optional

`__init__(name: str, mobility_mode: bool = False)`

Constructs all the necessary attributes for the PowNode object.

Parameters:
- **name** (*str*) – The node of the node to identify it.
- **mobility_mode** (*bool, optional*) – Indicates whether the node is mobile or stationary.

Examples

```
>>> node = PowNode(name='node1')
```

class node.**MidNode**(*name: str, mobility_mode: bool = False*)

Bases: `node.Node.Node`

A class that represents the module of a medium power node.

…

name

The node of the node to identify it.

Type: str

mobility_mode

Indicates whether the node is mobile or stationary.

Type: bool, optional

__init__(*name: str, mobility_mode: bool = False*)

Constructs all the necessary attributes for the MidNode object.

Parameters:
- **name** (*str*) – The node of the node to identify it.
- **mobility_mode** (*bool, optional*) – Indicates whether the node is mobile or stationary.

Examples

```
>>> node = PowNode(name='node1')
```

class node.**LowNode**(*name: str, mobility_mode: bool = False*)

Bases: `node.Node.Node`

A class that represents the module of a low power node.

…

name

The node of the node to identify it.

Type: str

mobility_mode

Indicates whether the node is mobile or stationary.

Type: bool, optional

__init__(*name: str, mobility_mode: bool = False*)

Constructs all the necessary attributes for the MidNode object.

Parameters:
- **name** (*str*) – The node of the node to identify it.
- **mobility_mode** (*bool, optional*) – Indicates whether the node is mobile or stationary.

Examples

```
>>> node = LowNode(name='node1')
```

class clientSelection.**ClientSelection**(*nodes: list, debug_mode: bool = False*)

A class that represents the basis of the client selection module.

…

nodes

The list of all nodes in the environment.

Type: list

debug_mode

Indicates if the debug mode is enabled or not.

Type: bool

get_nodes():

Return the list of all nodes.

get_debug_mode():

Return if the debug_mode is enabled.

__init__(*nodes: list, debug_mode: bool = False*)

Constructs all the necessary attributes for the ClientSelection object.

Parameters:
- **nodes** (*list*) – The list of all nodes in the environment.
- **debug_mode** (*bool, optional*) – Indicates if the debug mode is enabled or not.

get_debug_mode() → bool

Return if the debug_mode is enabled.

Returns: **debug_mode (bool)**
Return type: Indicates if the debug mode is enabled or not.

Examples

```
>>> clientSelection.get_debug_mode()
```

get_nodes() → list

Return the list of the nodes.

Returns: **nodes (list)**
Return type: the list of the nodes.

Examples

```
>>> clientSelection.get_nodes()
```

class clientSelection.**RandomClientSelection**(*nodes: list, K: float = 0.1, debug_mode: bool = False*)

A class that inherits the client selection module, which selects clients randomly.

…

nodes

The list of all nodes in the environment.

Type: list

`K`

the percentage of the selection.

Type: float

`debug_mode`

Indicates if the debug mode is enabled or not.

Type: bool

`random_client_selection():`

Returns a randomly selected list of clients with a percentage K.

`__init__(nodes: list, K: float = 0.1, debug_mode: bool = False)`

Constructs all the necessary attributes for the RandomClientSelection object.

Parameters:
- **nodes** (*list*) – The list of all nodes in the environment.
- **K** (*float*) – the percentage of the selection.
- **debug_mode** (*bool, optional*) – Indicates if the debug mode is enabled or not.

`random_client_selection() → list`

Return the list of selected nodes randomlu.

Returns: **selected_clients (list)**

Return type: the list of the nodes.

Examples

```
>>> randomClientSelection.random_client_selection()
```

class `clientSelection.ResourceClientSelection(nodes: list, K: float = 0.1, debug_mode: bool = False)`

A class that inherits the client selection module, which selects clients according to the strength of their resources.

…

`nodes`

The list of all nodes in the environment.

Type: list

`K`

the percentage of the selection.

Type: float

`debug_mode`

Indicates if the debug mode is enabled or not.

Type: bool

`resource_client_selection():`

Returns a list of clients selected according to their power ranking.

`__init__(nodes: list, K: float = 0.1, debug_mode: bool = False)`

Constructs all the necessary attributes for the ResourceClientSelection object.

Parameters:
- **nodes** (*list*) – The list of all nodes in the environment.
- **K** (*float*) – the percentage of the selection.
- **debug_mode** (*bool, optional*) – Indicates if the debug mode is enabled or not.

`resource_client_selection() → list`

Return the list of the selected nodes according to their power.

Returns:	**selected_clients (list)**
Return type:	the list of the nodes.

Examples

```
>>> resourceClientSelection.resource_client_selection()
```

`class consumptionModel.CPUModel(node: node.Node.Node)`

A class that represents the module of the processor consumption model of the nodes.

…

node

The node assigned to this model.

Type: Node

get_node(): Node

Return the node.

set_node():

Assign this consumption model to a node.

check_cpu():

Check if the node has reached the maximum CPU consumption level.

update_cpu():

Update the CPU consumption percentage of the node.

`__init__(node: node.Node.Node)`

Constructs all the necessary attributes for the CPUModel object.

Parameters: **node** (*Node*) – The node assigned to this model.

Examples

```
>>> cpuModel = CPUModel(node=node1)
```

`check_cpu() → bool`

Check if the node has reached the maximum CPU consumption level.

Returns:	**status (bool)**
Return type:	If the maximum level of consumption is reached or not.

Examples

```
>>> cpuModel.check_cpu()
```

`get_node() → node.Node.Node`

Return the instance of the node.

Returns: **nodes (Node)**

Return type: The node's instance.

> **Examples**
>
> ```
> >>> cpuModel.get_node()
> ```

set_node(*node: node.Node.Node*)

Assign this consumption model to a node.

Parameters: **node** (*Node*) – the node's instance.

> **Examples**
>
> ```
> >>> cpuModel.set_node(node1)
> ```

update_cpu(*cpu_usage*)

Update the CPU consumption percentage of the node.

Parameters: **cpu_usage** (*float*) – the new level of processor consumption.

> **Examples**
>
> ```
> >>> cpuModel.update_cpu(65)
> ```

class **consumptionModel.EnergyModel(*node: node.Node.Node*)**

A class that represents the module of the energy consumption model of the nodes.

…

node

The node assigned to this model.

Type: Node

get_node(): Node

Return the node.

set_node():

Assign this consumption model to a node.

consume_energy():

Consume a certain level of energy from the node according to its category.

check_battery():

Update the CPU consumption percentage of the node.

__init__(*node: node.Node.Node*)

Constructs all the necessary attributes for the EnergyModel object.

Parameters: **node** (*Node*) – The node assigned to this model.

> **Examples**
>
> ```
> >>> energyModel = EnergyModel(node=node1)
> ```

check_battery()

Check that the node 's battery is not depleted.

Examples

```
>>> cpuModel.check_battery()
```

consume_energy() → float

Consume a certain level of energy from the node according to its category.

Returns:	**new_energy (float)**
Return type:	The new energy level of the node.

Examples

```
>>> energyModel.consume_energy()
```

get_node() → node.Node.Node

Return the instance of the node.

Returns:	**nodes (Node)**
Return type:	The node's instance.

Examples

```
>>> energyModel.get_node()
```

set_node(*node: node.Node.Node*)

Assign this consumption model to a node.

Parameters:	**node** (*Node*) – the node's instance.

Examples

```
>>> energyModel.set_node(node1)
```

class **consumptionModel.MemoryModel(*node: node.Node.Node*)**

A class that represents the module of the memory consumption model of the nodes.

…

node

The node assigned to this model.

Type:	Node

get_node(): Node

Return the node.

set_node():

Assign this consumption model to a node.

check_memory():

Check if the node has reached the maximum Memory consumption level.

update_memory():

Update the Memory consumption percentage of the node.

`__init__(node: node.Node.Node)`

Constructs all the necessary attributes for the MemoryModel object.

Parameters: **node** (*Node*) – The node assigned to this model.

Examples

```
>>> memoryModel = MemoryModel(node=node1)
```

`check_memory() → bool`

Check if the node has reached the maximum Memory consumption level.

Returns: **status (bool)**

Return type: If the maximum level of consumption is reached or not.

Examples

```
>>> memoryModel.check_memory()
```

`get_node() → node.Node.Node`

Return the instance of the node.

Returns: **nodes (Node)**

Return type: The node's instance.

Examples

```
>>> memoryModel.get_node()
```

`set_node(node: node.Node.Node)`

Assign this consumption model to a node.

Parameters: **node** (*Node*) – the node's instance.

Examples

```
>>> memoryModel.set_node(node1)
```

`update_memory(memory_usage)`

Update the Memory consumption percentage of the node.

Parameters: **memory_usage** (*float*) – the new level of processor consumption.

Examples

```
>>> memoryModel.update_cpu(57)
```

class `consumptionModel.StorageModel(node: node.Node.Node)`

A class that represents the module of the storage consumption model of the nodes.

…

`node`

The node assigned to this model.

Type: Node

`get_node(): Node`

Return the node.

`set_node():`

Assign this consumption model to a node.

`check_storage():`

Check if the node has reached the maximum storage consumption level.

`add_to_storage():`

Add files to the node and thus fill the storage.

`__init__(node: node.Node.Node)`

Constructs all the necessary attributes for the StorageModel object.

Parameters: **node** (*Node*) – The node assigned to this model.

Examples

```
>>> storageModel = StorageModel(node=node1)
```

`add_to_storage(number_of_mega_bytes: float) → float`

Add files to the node and thus fill the storage.

Parameters: **number_of_mega_bytes** (*float*) – The size of the files added to the storage.

Examples

```
>>> storageModel.add_to_storage(658)
```

`check_storage() → bool`

Check if the node has reached the maximum Storage consumption level.

Returns: **status (bool)**
Return type: If the maximum level of consumption is reached or not.

Examples

```
>>> storageModel.check_storage()
```

`get_node() → node.Node.Node`

Return the instance of the node.

Returns: **nodes (Node)**
Return type: The node's instance.

Examples

```
>>> storageModel.get_node()
```

`set_node(node: node.Node.Node)`

Assign this consumption model to a node.

Parameters: **node** (*Node*) – the node's instance.

Examples

```
>>> storageModel.set_node(node1)
```

class network.**Network**(*nodes: list*, *network_number: int*, *debug_mode: bool = False*)

A class that represents the module of a network of nodes.

…

nodes

The list of all nodes in the environment.

Type: list

network_number

The network number.

Type: int

debug_mode

Indicates if the debug mode is enabled or not.

Type: bool

assign_ip_addresses():

Give an ip address to all nodes assigned to this network.

get_network_number():

Return the number of the network.

get_nodes():

Return the list of all nodes.

get_network_leader():

Return the leader of the network.

set_network_leader():

Assign a leader to the network.

__init__(*nodes: list*, *network_number: int*, *debug_mode: bool = False*)

Constructs all the necessary attributes for the Network object.

Parameters:

- **nodes** (*list*) – The list of all nodes in the environment.
- **network_number** (*int*) – The network number.
- **debug_mode** (*bool, optional*) – Indicates if the debug mode is enabled or not.

get_network_leader()

Return the leader of the network.

Returns: **network_leader (Node)**

Return type: the instance of the leader node of the network.

Examples

```
>>> network.get_network_leader()
```

get_network_number()

Return the number of the network.

Returns: **network_number (int)**
Return type: The network number.

Examples

```
>>> network.get_network_number()
```

`get_nodes() → list`

Return the list of the nodes.

Returns: **nodes (list)**
Return type: the list of the nodes.

Examples

```
>>> network.get_nodes()
```

`set_network_leader(network_leader)`

Assign a leader to the network.

Parameters: **network_leader** (*Node*) – The new node leader for the network.
Returns: **network_leader (Node)**
Return type: the instance of the leader node of the network.

Examples

```
>>> network.set_network_leader(node1)
```

`distribuedLearning.dist_learning(train_dataset, selected_clients: list, global_model, global_round: int) -> (<class 'float'>, <class 'list'>, <class 'dict'>, <class 'float'>)`

The function of distributed learning of nodes.

Parameters:
- **train_dataset** (*any*) – The training dataset used for distributed learning.
- **selected_clients** (*list*) – The selected clients.
- **global_model** (*any*) – The model global.
- **global_round** (*int*) – The current global round.

Returns:
- **loss_avg (float)** (*the avereage loss.*)
- **list_acc (list)** (*the full list of all accuracy.*)
- **clients_acc (dict)** (*the full list of clients accuracy as dict format.*)
- **energy (float)** (*The total energy consumed during distributed learning.*)

Examples

```
>>> loss_avg, list_acc, clients_acc, energy =
dist_learning(train_dataset=train_dataset,
selected_clients=selected_clients, global_model=global_model,
global_round=epoch)
```

By AICHE Mohamed
© Copyright 2022, AICHE Mohamed.