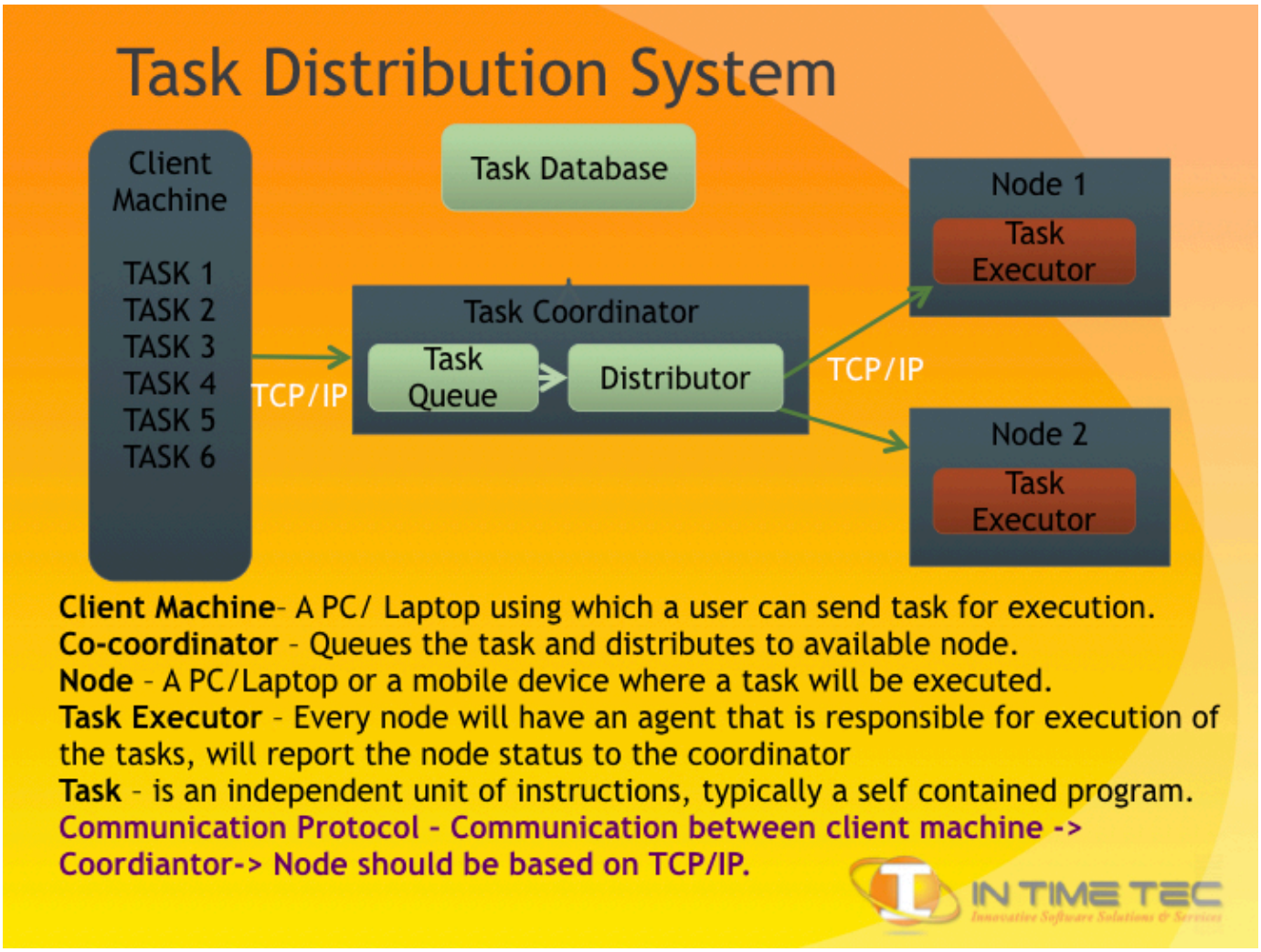
Overview


Task Distribution System
Client Machine
TASK 1
TASK 2
TASK 3
TASK 4
TASK 5
TASK 6
TCP/IP
Task Database
Task Coordinator
Task Queue
Distributor
TCP/IP
Node 1
Task Executor
Node 2
Task Executor
Client Machine- A PC/ Laptop using which a user can send task for execution.
Co-coordinator - Queues the task and distributes to available node.
Node - A PC/Laptop or a mobile device where a task will be executed.
Task Executor - Every node will have an agent that is responsible for execution of the tasks, will report the node status to the coordinator
Task - is an independent unit of instructions, typically a self contained program.
Communication Protocol - Communication between client machine -> Coordiantor-> Node should be based on TCP/IP.
IN TIME TEC

Task Distribution Interface

- C:>taskmgr queue “program1.exe”

queued, task ID 1

- C:>taskmgr query 1
- Status: queued or executing or completed
- C:\>taskmgr result 1

“Hello World 1”

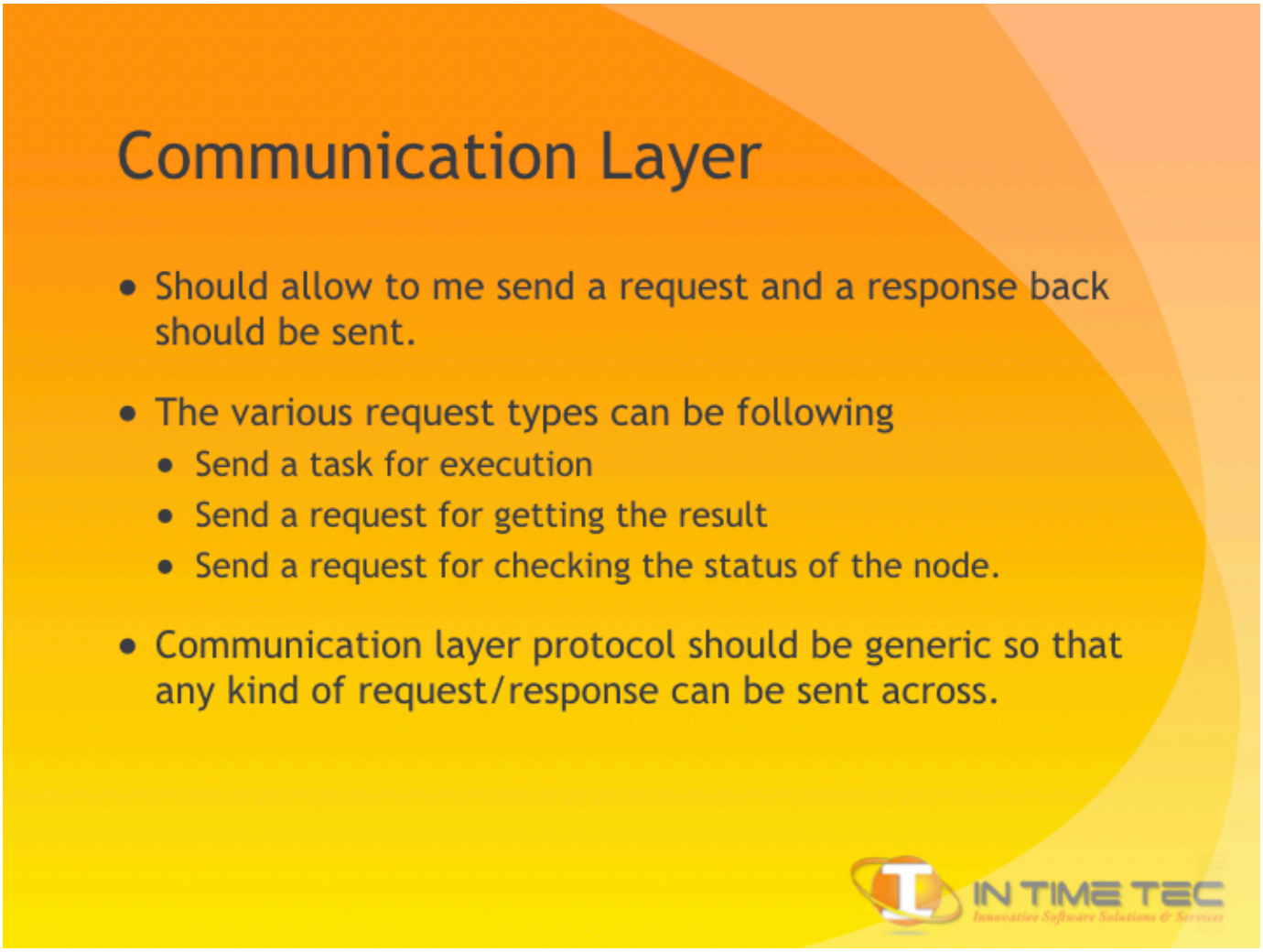
The program that is queued needs to be transferred the node where the execution needs to be performed.

List of nouns from the main problem statement

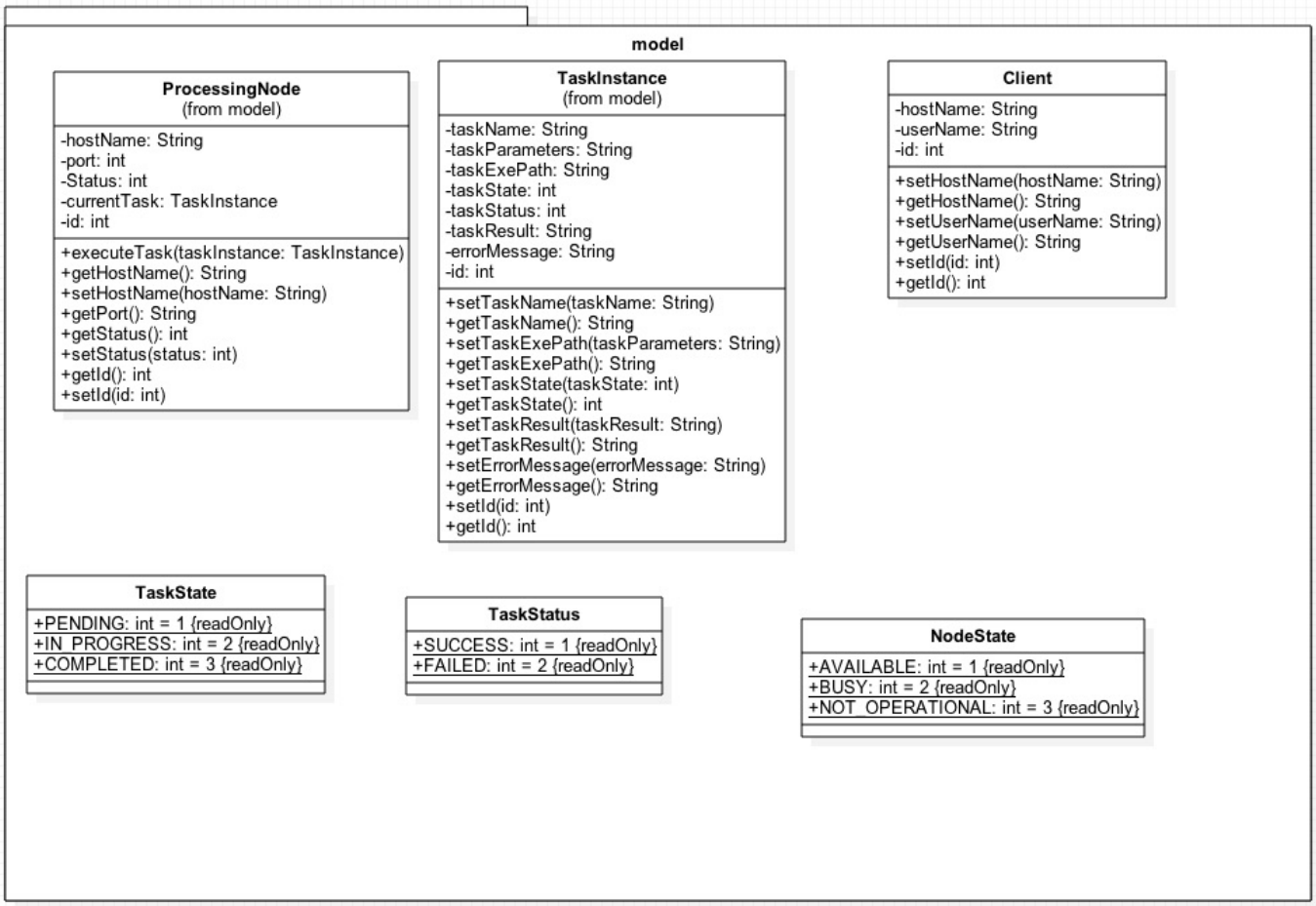
- Co-ordinator
- Client
- User
- Node (Mobile,Desktop)
- Task ?
- Agent (TaskExecutor)
- NodeStatus - Data
- Queue

Datbase Layer problem statement

- Database layer allows us to store/retrieve/ the task and it's parameters(Task Context).
- Database layer allows us to retrieve the task result
- Datbase layer allow us to store/retrieve the Node information(Node Context) and it's status.
- Database layer allows us to store user information.


Communication Layer
Should allow to me send a request and a response back should be sent.
The various request types can be following
Send a task for execution
Send a request for getting the result
Send a request for checking the status of the node.
Communication layer protocol should be generic so that any kind of request/response can be sent across.
IN TIME TEC

Core Models


model
ProcessingNode
(from model)
-hostName: String
-port: int
-Status: int
-currentTask: TaskInstance
-id: int
+executeTask(taskInstance: TaskInstance)
+getHostName(): String
+setHostName(hostName: String)
+getPort(): String
+getStatus(): int
+setStatus(status: int)
+getId(): int
+setId(id: int)
TaskInstance
(from model)
-taskName: String
-taskParameters: String
-taskExePath: String
-taskState: int
-taskStatus: int
-taskResult: String
-errorMessage: String
-id: int
+setTaskName(taskName: String)
+getTaskName(): String
+setTaskExePath(taskParameters: String)
+getTaskExePath(): String
+setTaskState(taskState: int)
+getTaskState(): int
+setTaskResult(taskResult: String)
+getTaskResult(): String
+setErrorMessage(errorMessage: String)
+getErrorMessage(): String
+setId(id: int)
+getId(): int
Client
-hostName: String
-userName: String
-id: int
+setHostName(hostName: String)
+getHostName(): String
+setUserName(userName: String)
+getUserName(): String
+setId(id: int)
+getId(): int
TaskState
+PENDING: int = 1 {readOnly}
+IN_PROGRESS: int = 2 {readOnly}
+COMPLETED: int = 3 {readOnly}
TaskStatus
+SUCCESS: int = 1 {readOnly}
+FAILED: int = 2 {readOnly}
NodeState
+AVAILABLE: int = 1 {readOnly}
+BUSY: int = 2 {readOnly}
+NOT_OPERATIONAL: int = 3 {readOnly}