Master - Minions Network Attached Storage

1. **BackGround:**  
   The project supplies a mechanism which allows taking advantage of external storage devices. As the IOT suggests, our system consists of interrelated computing devices, mechanical and digital machines provided with unique id and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. The end user will see all the entire external storage devices as one external server that will be used as an external storage device. Our system operates as a **NAS** (Network Attached Storage) server which is a file-level computer data storage server connected to a computer network providing data access to a heterogeneous group of clients.  
   We use **RAID 01** technique to equivalently stripe the data across multiple disks and increase reliability by creating a copy of each data block that was written within another disk.

The **Master - Minions** is a model of asymmetric communication where one device (Master) controls multiple external devices (Minions).  
We’re using the **NBD** (Network Block Device) to send requests via the kernel back to the user space. A specific folder should be chosen and be mounted to the NBD. Once that happens, and the program is running, each change within the mounted folder will be translated as a unique request and will be transferred to the external storage devices in a form of RAID01 technique as explained before.

1. **Purpose:**

* Educational purposes:

1. Design Patterns

* Factory
* Singleton
* Dispatcher
* Reactor
* Subject - Observer
* ThreadPool
* Command

1. Networking
2. Serializer

* Project purposes:

1. Using external storage devices to store data externally by taking advantage of already existing smart devices.
2. Creating a default backup mechanism for the entire transferred data.
3. Working in parallel to improve performance.
4. **Attributes:**
5. EventDriven
6. Configurability:  
   The architure of our system can be modified as needed. For instance, the amount of minions(external storage devices) can be changed.
7. Fault Tolerance:  
   We used a WatchDog software component to monitor each process in our system.
8. Plug and Play:  
   Our program is extensible. New features can be added during run-time.
9. **Verification**Our program will be verified by using an external Raspberry Pi as a storage device.
10. **UML:**

<https://app.diagrams.net/#G123-MJ7NtmN65iq9IsoiNiZT14vEfH59n>

1. **SequenceDiagram:**  <https://app.diagrams.net/#G1LIxtvTgyLrss_5W9JU6RfKkFC0aWvdQc>

Define The Protocol Between the Master and the Minion

Every Minion / Master have a unique id

Flow Description:

* Master sends every 1 minute broadcast request
* Minion answer only on the following scenarios:
* If minion is not attached to any master
* If minion attached to the master ID

**BroadCast Request (Master to minion)**

Identify which available

|  |  |  |
| --- | --- | --- |
| Name | Size | Notes |
| Master Id | 4 bytes |  |
| Type | 1 Byte | ‘b’ |

**BroadCast Answer (Minion to Master)**

Identify which available

|  |  |  |
| --- | --- | --- |
| Name | Size | Notes |
| Minion Id | 4 bytes |  |
| Type | 1 Byte | ‘b’ |

**Attach**

**Only when the master decides that it wants to attach the minion it will send the following request.**

**Request (Master to minion):**

|  |  |  |
| --- | --- | --- |
| Name | Size | Notes |
| MasterID | 4 bytes |  |
| Type | 1 Byte | ‘a’ |
| Minion Id | 4 bytes |  |

**Answer (Minion to Master):**

|  |  |  |
| --- | --- | --- |
| Name | Size | Notes |
| Minion Id | 4 bytes |  |
| Type | 1 Byte | ‘a’ |
| Status | 1 Bytes | ACK: 0x06 / NACK:0x15 |

**Detach**

**Request (Master to minion):**

|  |  |  |
| --- | --- | --- |
| Name | Size | Notes |
| MasterID | 4 bytes |  |
| Type | 1 B  yte | ‘d’ |
| Minion Id | 4 bytes |  |

**Answer (Minion to Master):**

|  |  |  |
| --- | --- | --- |
| Name | Size | Notes |
| Minion Id | 4 bytes |  |
| Type | 1 Byte | ‘d’ |
| Status | 1 Bytes | ACK: 0x06 / NACK:0x15 |