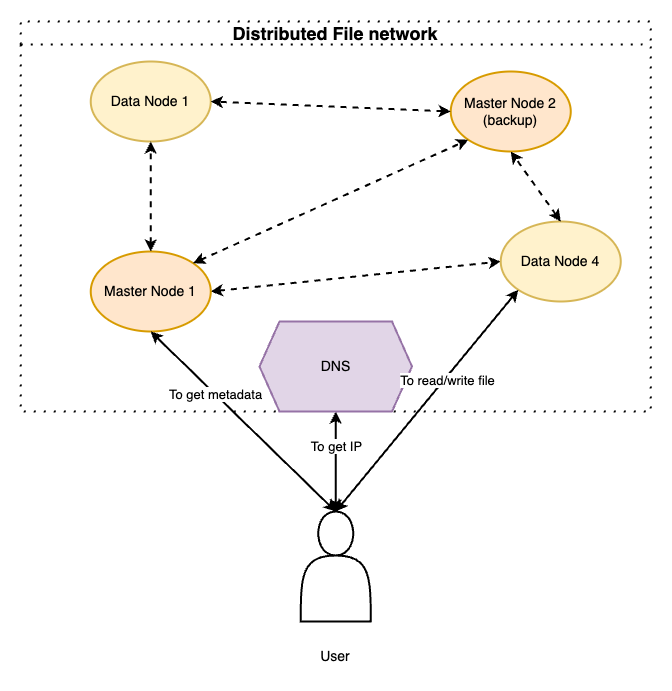
Project description

# 1. Overview

This file distributed system allows the user to send/receive file from a distributed file system. In this system, each file will have at least 2 replicas stored in two separate nodes.



## 1.1. Intro

* Architecture: 2 types of nodes: Master and Data node. Master keeps files replica and controls. Data node keeps the files replica only. DNS server keeps the IP of the current Master node
* Communication: bare socket-based solution
* State: During working, the current Master node and the backup Master node always sync the metadata.
* Coordination: DNS server is used to get the IP of the current Master node. When the current Master is down, there is an algorithm that makes the backup Master node and the current.
* Data consistency: Master node only permits 2 scenarios for each file:

+ multiple read

+ zero read, single write

* Replication: Each file has at least 2 replicas stored in different Data nodes. After one Data node successfully receives a file uploaded from client, the **Replication process** begins.
* Fault tolerance: With Data node and Master node, each has a distinct **Fault tolerance strategy** when it is down.

+ for Data node: check section 3.4.b.

+ for Master node: check section 3.4.b.

# 2. Functional requirements

* User:
  + can send file to specific node within the network, the file will later be replicated to another chosen node
  + can upload and read file
* Nodes can communicate with each other to exchange:
* Files
* Health status (heartbeat)
* The metadata of which node is holding which file
* When a node is down, other nodes must exchange the replica of the files held by the down one to ensure the replica of every file must be at least 2

# 3. System architecture

## 3.1. User

User is a program which can send TCP requests to Master and Data node in the system. User connects to DNS to get the IP of the current Master node.

**Assumption(s)**:

* We assume user has 2 permissions:
* read file
* write file

## 3.2. DNS

DNS stores the IP of the current Master node.

**Assumption(s)**:

* We assume that DNS is never down.

## 3.2. Master node

Our system shares similarities with **RAFT algorithm**.

Master node has several roles:

* Store files’ replica
* keep track of which file is located on which data node
* keep track of Data Node alive status via **heartbeat**
* trigger replication process when a node is down

There are 2 Master nodes, but only one of them does the roles listed above. The other Master node is just backup. During working, the actual Master node exchanges the metadata to the backup to ensure when it is down, the backup can replace and become the working Master node.

## 3.3. Data node

Data node has only one target: keep the files’ replica.

Periodically, it sends heartbeat message to Master node to keep track health status.

During **Replication process**, Data node receives the IP address of another data node to send data.

## 3.4. File distributed system

### a. Consistency guarantee

For each file, Master node permits 2 scenarios from clients:

* multi read: concurrently read by multiple clients
* no-read-one-write: during a client is writing (uploading) the file, no other client can read it

### b. Fault tolerance

There are 2 distinct scenarios of fault tolerance for Master and Data.

* For Master fault tolerance: When the current Master node is down, it can be only determined by the backup Master node via **2 consecutive non-response heartbeats**. In this case, the backup Master node triggers the **Master node fault tolerance process** (described in section 3.4.g).
* For Data node fault tolerance: When the current Master node detects a Data node doesn’t response 2 consecutive heartbeats, for each file whose the replica is held by the down Data node, the current Master node chooses an alternative Data node and triggers the **Replication process**.

**Assumption(s)**:

* Due to time limit, we only implement **Replication process** but in the final, we will describe the algorithm for **Master node fault tolerance process**

### c. Coordination

DNS server is used to store the IP of the current Master node. All other Data nodes after initializing and Client, which want to know the IP of the current Master node, must connect to the DNS.

When the current Master node is down and the backup Master node becomes the current Master node, the new Master node has responsibility to update its IP to the DNS server.

### d. Replication

Replication is the process triggered by the current Master node to force one Data node to send specific file to another Data node. This is triggered in either:

* The new file is newly written (uploaded) by client
* A Data node is down

Regarding the **selection of which Data node will receive the replica**, the current Master chooses which Data node has the **largest remaining available size**. The remaining size is calculated as follow:

* Assume each Data node has 100 available slots in the beginning => After the Data node receives a file, the number of slots is deduced by 1.

### e. Packet format

In the payload, if it contains the file name and the binary file at the same time, this 2 information are separated by character ‘**||**’.

The rule is as follow:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **packetName** | **payloadSize** | **Payload** | **Purpose** | **From -> To** |
| Heartbeat | 0 | [] | Check health status of nodes (heartbeat) | Current Master -> Data node  Current Master -> Backup Master  Backup Master -> Current Master |
| HeartbeatAck | n | [<node\_id>] | Heartbeat ACK | Data node -> Current Master  Current Master -> Backup Master  Backup Master -> Current Master |
| 3 | n | [<file name>, <Data Node IP><port>] | Current Master requests Data node to send a replica of a specific file to another Data node | Current Master -> Data node |
| 4 | n | [<file name>||<binary>] | Transmit file replica | Data node -> Data node |
| 5 | n | [<file name>] | 2 Data nodes notify the Master the replication process of specific file is done (ACK of replication) | Data node -> Current Master |
| AskIP | 2 | [<port>] | Ask the IP of the current Master node  Payload | Client -> DNS |
| AskIPAck | n | [<Master address’s IP><Master address’s port>] | DNS responses the current Master IP. If Master IP is available, 6 consecutive bytes in payload indicate the IP and port of Master. Otherwise, payload is empty. | DNS -> Client |
| 8 | n | [<listen\_port\_client><file name><READ/WRITE] | Client asks to read/write specific file | Client -> Current Master  Client -> Data node |
| 9 | n | [<Data Node IP><port>] | Master responses which Data node is keeping the file replica/ready to receive file | Current Master -> Client |
| 10 | n | [<file name><binary>] | Client writes (uploads) file | Client -> Data node |
| 11 | n | [<file name><file binary>] | Data node response file binary | Data node -> Client |
| 12 | n | [<file name>||READ/WRITE] | Data node notifies the Master the reading/writing process from client is done | Data node -> Current Master |
| 13 | n | [<state>] | Current Master node synchronizes its state with the backup Master | Current Master -> Backup Master |
| 14 | 0 | [] | ACK of state synchronization | Backup Master -> Current Master |
| Notify | n | [<role><IP><port>] | Either of 3 scenarios:  i) new Master node notifies to DNS  ii) new Data node notifies to current Master  iii) new Master notifies to current Master | Master -> DNS  Master -> Master  Data node -> Master |

### f. State synchronization

After 30 seconds, the current Master node synchronizes its state with the backup Master node. The state is the hash map containing the info: for each file, which Data nodes are holding its replica.

### g. Flows

|  |  |  |
| --- | --- | --- |
| **Name** | **In-charge components** | **Flow** |
| Initialization | DNS, Master, Data | For Master:   1. Master sends Notify to DNS   For Data:   1. Data sends AskIP to DNS 2. DNS replies by AskIP containing the socket address of current Master 3. Data sends Notify to Master |
| Read data | Client;  DNS;  Current Master;  1 Data node | 1. Client connects to DNS to ask the IP of current Master *[packet 6]*  2. DNS responses the IP *[packet 7]*  3. Client connects to the current Master to ask to read specific file *[packet 8]*  4. Master responses the IP of the Data node is holding file replica *[packet 9]*  5. Client connects to Data node to get file *[packet 8]*  6. Data node responses the file binary *[packet 11]*  7. Data node sends ACK for finishing reading process *[packet 12]* |
| Write data | Client;  DNS;  Current Master;  2 Data nodes | 1. Client connects to DNS to ask the IP of current Master *[packet 6]*  2. DNS responses the IP *[packet 6]*  3. Client connects to the current Master to ask to write specific file *[packet 8]*  4. Master responses the IP of the Data node is ready to receive file *[packet 9]*  5. Client connects to Data node to write file *[packet 10]*  6. When client finishes writing, Data node notifies the Master node the writing process is done *[packet 12]*  7. The current Master node selects the suitable Data node different from the one which is newly received file from the client and triggers **Replication process**. |
| Heartbeat | Current Master ; Backup Master  all Data nodes | Periodically, each Data node sends a heartbeat to report the health status to Master.  1. Node **A** sends heartbeat to **B** *[packet 1]*  2. **B** sends ACK back to **A** *[packet 2]* |
| Fault tolerance on Data node | Current Master;  2 Data nodes | 1. After 2 heartbeats from current Master but the Data node (denote **A**) doesn’t response, the current Master looks up for each file **X** held by **A**, which Data node (denote **B**) is holding a replica  2. For each **X** and **B**, the current Master triggers the **Replication process**. |
| Fault tolerance on Master node | Current Master and backup Master | 1. After 2 heartbeats from backup Master but the current Master doesn’t response, the backup Master notifies all Data nodes that it now is becoming the current Data node  2. New current Master node updates its IP to DNS server |
| Replication | Current Master; 2 Data Node | Assume the current Master wants to replicate the file **X** in Data node **A**.  1. Given a file to replicate, current Master selects the suitable Data node to store (denote as **B**)  2. Current Master notifies **A** to send file **X** to **B** *[packet 3]*  3. **A** sends **X** to **B** *[packet 4]*  4. Both **A** and **B** send ACK to current Master *[packet 5]* |
| State synchronization | Current Master node and backup Master node | 1. After 30 seconds, the current Master Node synchronizes its current state to the backup Master *[packet 13]*  2. After receiving, the backup Master sends ACK *[packet 14]* |