Chapter 7: Failure Detection

In a distributed system, when a client sends a request to a server, several things can go wrong. If no response comes back after some time, it is impossible for the client to know the exact cause of the failure. The server could be slow, it could have crashed, or there could be a network issue that prevented the request or the response from being delivered.

Because of this uncertainty, it is **not possible to build a perfect failure detector**. The best a client can do is make an educated guess, after a certain amount of time has passed, that the server is unavailable.

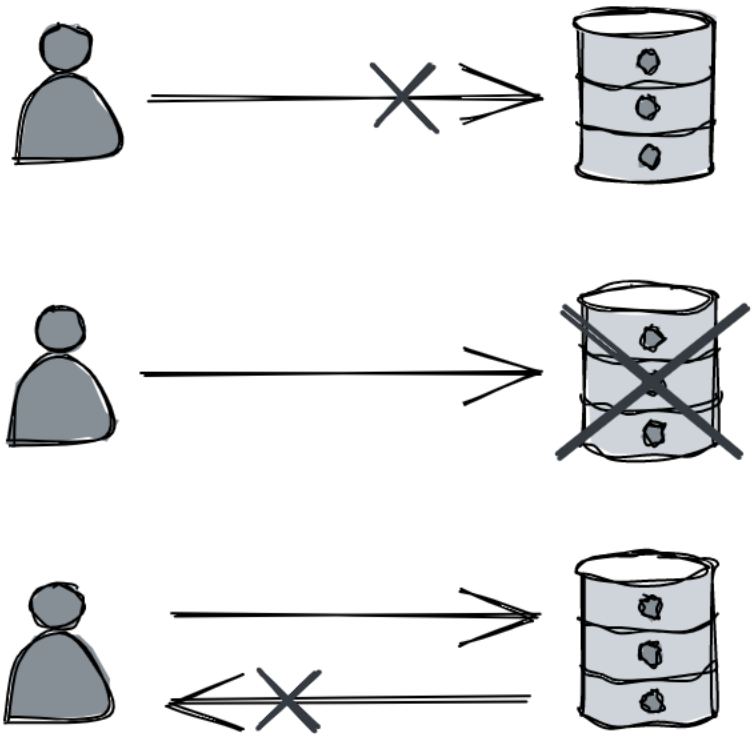

Figure 1: The client can't tell whether the server is slow, it crashed or a message was delayed/dropped because of a network issue.

Timeouts

The most common mechanism for detecting failures is a **timeout**.

- A client configures a timeout to trigger if it has not received a response from the server after a specified amount of time.
- When the timeout triggers, the client considers the server to be unavailable.
- At this point, the client can either throw an error or retry the request.

The main challenge is choosing the right duration for the timeout:

- **If the delay is too short**, the client might wrongly assume the server is unavailable when it is just slow.
- **If the delay is too long**, the client might waste valuable time waiting for a response that will never arrive.

Proactive Failure Detection

Instead of waiting for a request to fail, a process can proactively monitor other processes to check their availability. The two primary methods for this are **pings** and **heartbeats**.

These methods are generally used for processes that interact frequently and where an immediate action must be taken as soon as a process becomes unreachable. In other situations, detecting failures reactively at communication time is often sufficient.

Pings

- A process periodically **sends** a `ping` request to another process to check if it is still available.
- The sender expects to receive a response to the ping within a specific time frame.
- If no response is received, a timeout triggers, and the destination process is

considered unavailable.

- The process will continue to send pings to the destination to detect if and when it comes back online.

Heartbeats

- A process periodically **sends** a `heartbeat` message to another process.
- The *receiving* process expects to get a heartbeat within a specific time frame.
- If the destination does not receive a heartbeat within that period, it triggers a timeout and considers the sending process to be unavailable.
- If the sending process later comes back to life and resumes sending heartbeats, it will eventually be considered available again.