[Circuit Breaker]

↳ To avoid cascading failures

What is cascading failures.

↓

Failure of one service leads to failure of other services

The Circuit Breaker design pattern is used to stop the request and response process if a service is not working, as the name suggests.

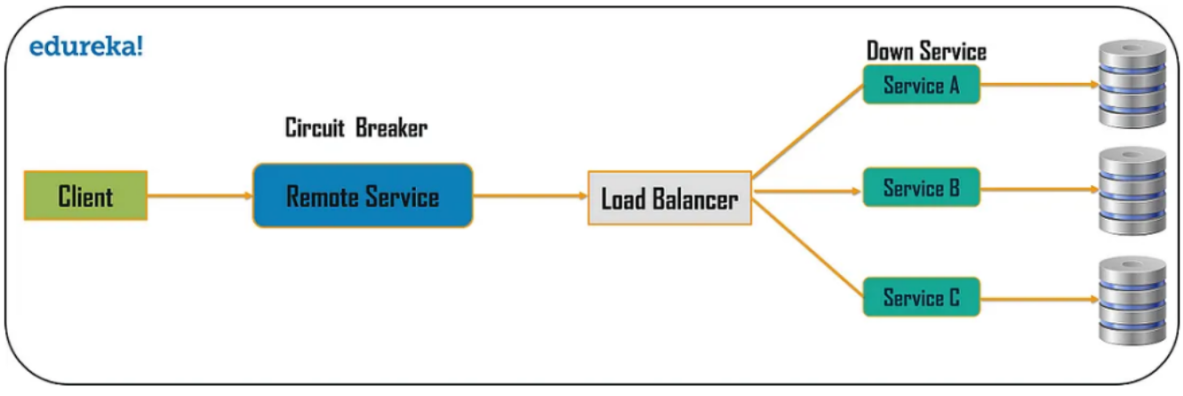


Figure 1: Circuit Breaker Pattern

🌟 As an example, assume a consumer sends a request to get data from multiple services. But, one of the services is unavailable due to technical issues. There are mainly two issues you will face now.

- First, because the consumer will be unaware that a particular service is unavailable (failed), so the requests will be sent to that service continuously.
- The second issue is that network resources will be exhausted with low performance and user experience.

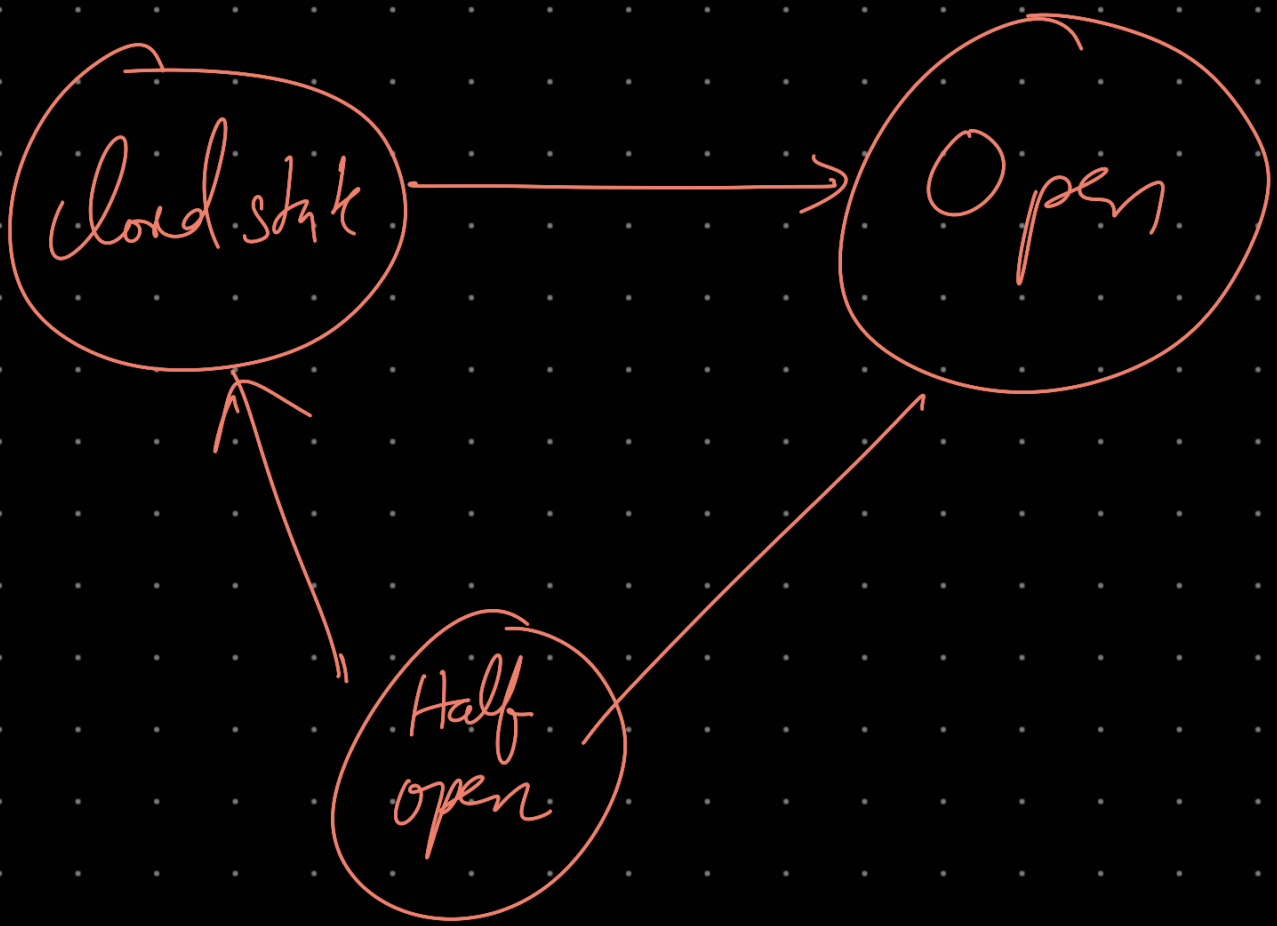
🌟 You can leverage the Circuit Breaker Design Pattern to avoid such issues. The consumer will use this pattern to invoke a remote service using a proxy. This proxy will behave as a circuit barrier.

🌟 When the number of failures reaches a certain threshold, the circuit breaker trips for a defined duration of time.

🌟 During this timeout period, any requests to the offline server will fail. When that time period is up, the circuit breaker will allow a limited number of tests to pass, and if those requests are successful, the circuit breaker will return to normal operation. If there is a failure, the time out period will start again.

States of the Circuit Breaker

1. **Closed State**:
   - The system operates normally.
   - All requests are sent to the external service/component.
   - If a certain number of failures (threshold) occur, the circuit breaker transitions to the **Open State**.
2. **Open State**:
   - The circuit breaker blocks requests to the failing service/component.
   - It immediately returns a fallback response or error.
   - This prevents overloading the service and allows it to recover.
3. **Half-Open State**:
   - After a predefined timeout, the circuit breaker enters a test phase.
   - A limited number of requests are allowed to pass through to the service.
   - If these requests succeed, the circuit breaker transitions back to **Closed State**.
   - If they fail, it transitions back to **Open State**.



Use Cases

- **Microservices**: To handle downstream service failures.
- **APIs**: When interacting with third-party APIs.
- **Databases**: To avoid overloading a failing database.