Sessions and Events

This lesson will explain Chubby sessions and what different Chubby events are.

We'll cover the following

What is a Chubby session?#

A Chubby session is a relationship between a Chubby cell and a Chubby client.

- It exists for some interval of time and is maintained by periodic handshakes called **KeepAlives**.
- Client's handles, locks, and cached data only remain valid provided its session remains valid.

Session protocol#

- Client requests a new session on first contacting the master of Chubby cell.

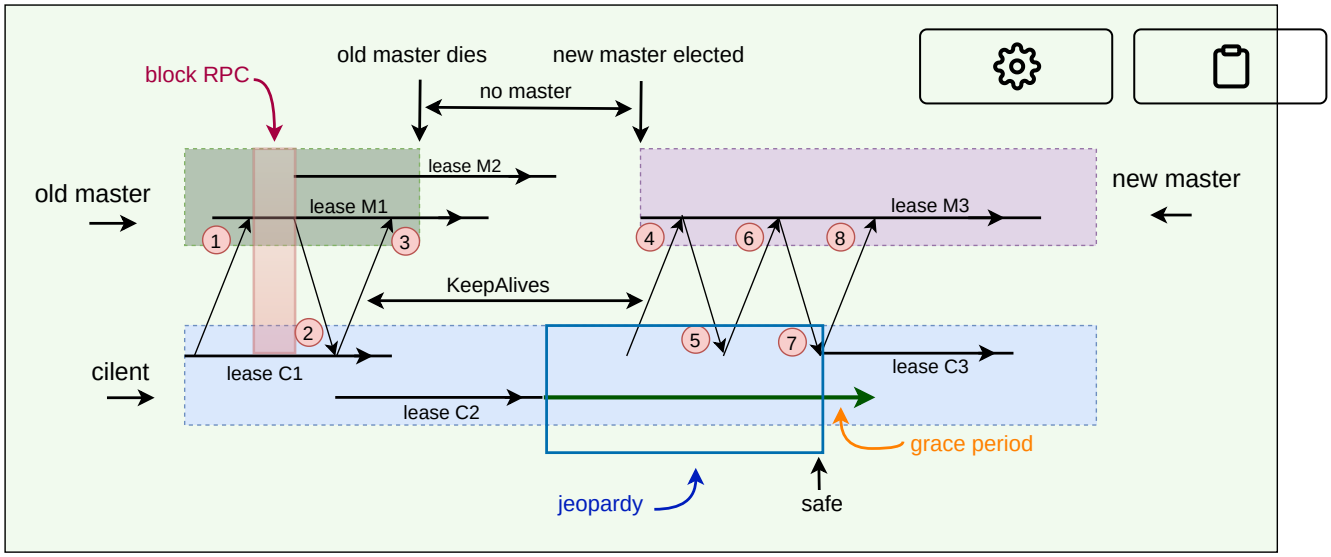
- A session ends if the client ends it explicitly or it has been idle. A session is considered idle if there are no open handles and calls for a minute.
- Each session has an associate lease, which is a time interval during which the master guarantees to not terminate the session unilaterally. The end of this interval is called ‘session lease timeout.’
- The master advances the ‘session lease timeout’ in the following three circumstances:
    - On session creation
    - When a master failover occurs
    - When the master responds to a KeepAlive RPC from the client

What is KeepAlive?#

KeepAlive is basically a way for a client to maintain a constant session with Chubby cell. Following are basic steps of responding to a KeepAlive:

- On receiving a KeepAlive (step “1” in the diagram below), the master typically blocks the RPC (does not allow it to return) until the client’s previous lease interval is close to expiring.
- The master later allows the RPC to return to the client (step “2”) and thus informs the client of the new lease timeout (lease M2).
- The master may extend the timeout by any amount. The default extension is 12s, but an overloaded master may use higher values to reduce the number of KeepAlive calls it must process. Note the difference between the lease timeout of the client and the master (M1 vs. C1 and M2 vs. C2).
- The client initiates a new KeepAlive immediately after receiving the previous reply. Thus, the client ensures that there is almost always a KeepAlive call blocked at the master.

In the diagram below, thick arrows represent lease sessions, upward arrows are KeepAlive requests, and downward arrows are KeepAlive responses. We will discuss this diagram in detail in the next two sections.



Client maintaining a session with Chubby cell through KeepAlive

Session optimization#

Piggybacking events: KeepAlive reply is used to transmit events and cache invalidations back to the client.

Local lease: The client maintains a local lease timeout that is a conservative approximation of the master's lease timeout.

Jeopardy: If a client's local lease timeout expires, it becomes unsure whether the master has terminated its session. The client empties and disables its cache, and we say that its session is in jeopardy.

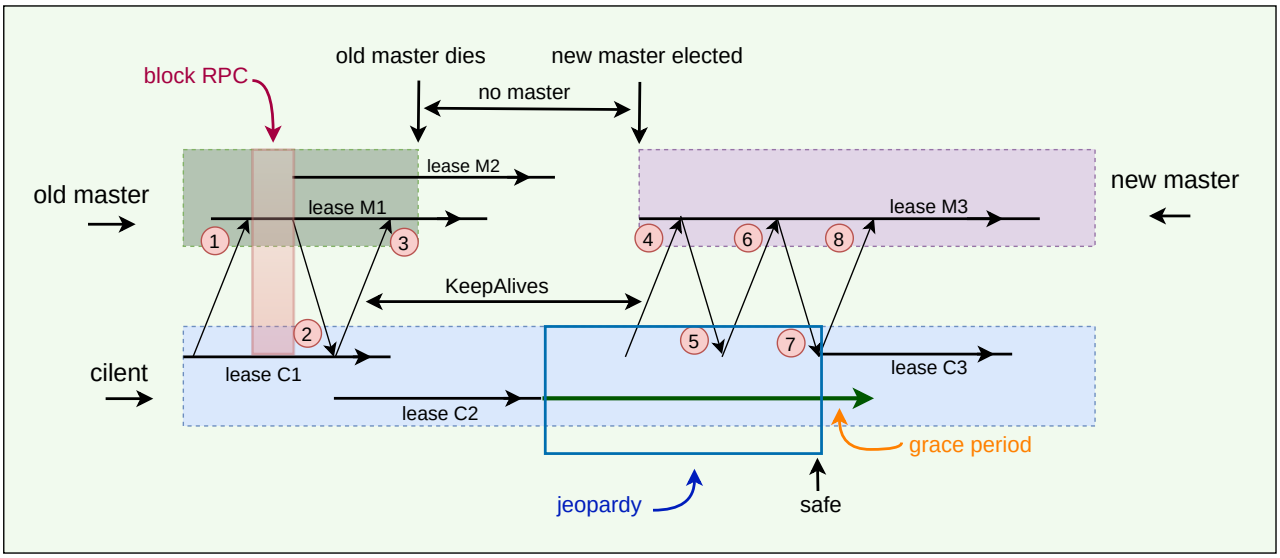
Grace period: When a session is in jeopardy, the client waits for an extra time called the grace period - 45s by default. If the client and master manage to exchange a successful KeepAlive before the end of client's grace period, the client enables its cache once more. Otherwise, the client assumes that the session has expired.

Failovers#

The failover scenario happens when a master fails or otherwise loses membership. Following is the summary of things that happen in case of a master failover:

- The failing master discards its in-memory state about sessions, handles, and locks.
- Session lease timer is stopped. This means no lease is expired during the time when the master failover is happening. This is equivalent to lease extension.
- If master election occurs quickly, the clients contact and continue with the new master before the client's local lease expires.
- If the election is delayed, the clients flush their caches (= jeopardy) and wait for the "grace period" (45s) while trying to find the new master.

Let's look at an example of failover in detail.



Chubby master failover

1. Client has lease M1 (& local lease C1) with master and pending KeepAlive request.
2. Master starts lease M2 and replies to the KeepAlive request.

3. Client extends the local lease to C2 and makes a new KeepAlive call. Master dies before replying to the next KeepAlive. So, no new leases can be assigned. Client's C2 lease expires, and the client library flushes its cache and informs the application that it has entered jeopardy. The grace period starts on the client.
4. Eventually, a new master is elected and initially uses a conservative approximation M3 of the session lease that its predecessor may have had for the client. Client sends KeepAlive to new master (4).
5. The first KeepAlive request from the client to the new master is rejected (5) because it has the wrong master epoch number (described in the next section).
6. Client retries with another KeepAlive request.
7. Retried KeepAlive succeeds. Client extends its lease to C3 and optionally informs the application that its session is no longer in jeopardy (session is in the safe mode now).
8. Client makes a new KeepAlive call, and the normal protocol works from this point onwards.
9. Because the grace period was long enough to cover the interval between the end of lease C2 and the beginning of lease C3, the client saw nothing but a delay. If the grace period was less than that interval, the client would have abandoned the session and reported the failure to the application.

← Back

Locks, Sequencers, and Lock-delays

Next →

Master Election and Chubby Events

Mark as Completed

Report an Issue