1. Network failure

(a) Packet delay

```
Client 1                        Server 1                        Client 2
RPC_lock_acquire();
                                                                RPC_lock_acquire();
                                ...grant the lock to C1
...lock acquired
RPC_append_file('A'); (delayed)

                                ...lock times out
                                ...grant the lock to C2
                                                                 ...lock acquired
                                (receive the delayed file_append)
                                ...refuse file_append from C1
```

Expected Result: Client 1 receives `ERROR: LOCK_EXPIRED` as Client 2 has already acquired the lock, and 'A' is not written to the file.

(b) Packet drop

- Client packet loss

```
Client 1                        Server 1                        Client 2
RPC_lock_acquire(); (lost)                                      RPC_lock_acquire();
                                ...grant the lock to C2
                                                                ...lock acquired
                                                                RPC_append_file('B');
                                ...append 'B'
                                                                ...'B' appended
                                                                RPC_lock_release();
                                ...release the lock for C2

retry: RPC_lock_acquire()
                                ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                                ...append 'A'
...'A' appended
```

Expected Result: Client 1 is able to acquire the lock through the retry mechanism, and the file contains "BA".

- Server packet loss

```
Client 1                        Server 1                        Client 2
RPC_lock_acquire();
                                                                RPC_lock_acquire();
                                ...grant the lock to C1 (lost)
```

```
retry: RPC_lock_acquire()
                                ...inform C1 it has the lock
...lock acquired
RPC_append_file('A');
                                ...append 'A'
...'A' appended
RPC_lock_release();
                                ...release the lock for C1
                                ...grant the lock to C2
                                                                ...lock acquired

RPC_append_file('B');
                                ...append 'B'
                                                                ...'B' appended
                                                                RPC_lock_release();
                                ...release the lock for C2
```

Expected Result: Client 1 queries the server with retry, finds out that it holds the lock, and sends file_append request; the file contains "AB".

(b) Duplicated packets

- lock_release

```
Client 1                        Server 1                        Client 2
RPC_lock_acquire();
                                                                RPC_lock_acquire();
                                ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                                ...append 'A'
...'A' appended
RPC_lock_release(); (delayed)

retry: RPC_lock_release();
                                ...release the lock for C1
...lock released                ...grant the lock to C2
                                                                ...lock acquired
                                (ignore the duplicated lock_release)
                                                                RPC_append_file('B');
                                ...append 'B'
                                                                ...'B' appended
                                                                ...wait for a while
RPC_lock_acquire();
                                                                RPC_append_file('B');
                                ...append 'B'
                                                                ...'B' appended
                                                                RPC_lock_release();
                                ...release the lock for C2
                                ...grant the lock to C1
...lock acquired
RPC_append_file('A');
```

```
                                ...append 'A'
...'A' appended
```

Expected Result: Client 1 does not mistakenly release the lock for Client 2, and it reacquries the lock after Client 2 releases it; the file contains "ABBA".

(b) Combined network failures

- Multiple packet drops & duplicated file_append

```
Client 1                        Server 1                        Client 2
RPC_lock_acquire();
                                                                RPC_lock_acquire();
                                ...grant the lock to C1
...lock acquired
RPC_append_file('1');
                                ...append '1'
...'1' appended
RPC_append_file('A'); (lost)

retry: RPC_append_file('A');
                                ...append 'A'
                                ...(reponse packet loss)

retry: RPC_append_file('A');
                                ...inform C1 file_append success
...'A' appended
RPC_lock_release();
                                ...release the lock for C1
                                ...grant the lock to C2
                                                                ...lock acquired

RPC_append_file('B');
                                ...append 'B'
                                                                ...'B' appended
```

Expected Result: Client 1 completes two file_append requests despite packet losses on both client and server sides; Client 2 then acquires the lock; the file contains "1AB," demonstrating idempotency.

2. Client fails/stucks

(a) Stucks before editing the file

```
Client 1                        Server 1                        Client 2
RPC_lock_acquire();
                                                                RPC_lock_acquire();
                                ...grant the lock to C1
...lock acquired

(...garbage collection happens)
                                ...lock times out (C1)
```

```
                              ...grant the lock to C2
                                                                ...lock acquired

RPC_append_file('B');
(...GC ends)                  ...append 'B'
RPC_append_file('A');                                           ...'B' appended
                              ...refuse file_append from C1
...ERROR: LOCK_EXPIRED                                          ...wair for a while

RPC_append_file('B');
                              ...append 'B'
RPC_lock_acquire();                                             ...'B' appended
                                                                RPC_lock_release();
                              ...release the lock for C2
                              ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                              ...append 'A'
...'A' appended
RPC_append_file('A');
                              ...append 'A'
...'A' appended
```

Expected Result: Client 2 acquires the lock after Client 1's lock ownership expires; Client 1 receives `ERROR: LOCK_EXPIRED` after its long pause and needs to reacquire the lock; the file contains "BBAA" (with Client 2's appended data appearing first).

(b) Stucks after editing the file

```
Client 1                      Server 1                      Client 2
RPC_lock_acquire();
                                                            RPC_lock_acquire();
                              ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                              ...append 'A'
...'A' appended
(...garbage collection happens)
                              ...lock times out (C1)
                              ...(option1) remove 'A'
                              ...grant the lock to C2
                                                            ...lock acquired

RPC_append_file('B');
(...GC ends)                  ...append 'B'
RPC_append_file('A');                                       ...'B' appended
                              ...refuse file_append from C1
...ERROR: LOCK_EXPIRED                                      ...wait for a while

RPC_append_file('B');
                              ...append 'B'
```

```
RPC_lock_acquire();                                          ...'B' appended
                                                             RPC_lock_release();
                              ...release the lock for C2
                              ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                              ...append 'A'
...'A' appended
RPC_append_file('A');
                              ...append 'A'
...'A' appended
```

Expected Result: Client 1's unfinished requests are aborted: 'A' is removed from the file; Client 2 gets the lock after timeout and appends 'B' twice; Client 1 comes back and redo its critical section; the file contains "BBAA".

NOTE: The workflow above only shows one option (option1) to guarantee the atomicity of clients' critical sections, you can also implement other solutions (e.g., transaction-based) as long as you can guarantee the two 'A's of client 1 are appended together.

3. Single server fails

(a) Lock is free

```
Client 1                          Server 1
RPC_lock_acquire();
                                  ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                                  ...append 'A'
...'A' appended
RPC_append_file('A');
                                  ...append 'A'
...'A' appended
RPC_lock_release();
                                  ...release the lock for C1
...lock released
                                  (x) Node crashes
RPC_lock_acquire();

                                  (v) Recovers
retry: RPC_lock_acquire();
                                  ...grant the lock to C1
RPC_append_file('1');
                                  ...append '1'
...'1' appended
RPC_lock_release();
                                  ...release the lock for C1
...lock released
```

Expected Result: Server is able to remember the previous commited data "AA", and it continues to accept clients' requests after recovery, thus Client 1 can acquire the lock again and append '1' to the file; the file contains "AA1".

(b) Lock is held

```
Client 1                  Server 1                  Client 2
RPC_lock_acquire();
                                                    RPC_lock_acquire();
                          ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                          ...append 'A'
...'A' appended
RPC_lock_release();
                          ...release the lock for C1
                          ...grant the lock to C2
                                                    ...lock acquired
                                                    RPC_append_file('B');
                          ...append 'B'
                          (x) Node crashes          ...'B' appended
                                                    RPC_append_file('B');

                          (v) Recovers              retry: RPC_append_file('B');
                          ...append 'B'
                                                    ...'B' appended
RPC_lock_acquire();
                                                    RPC_lock_release();
                          ...release the lock for C2
                          ...grant the lock to C1
...lock acquired
RPC_append_file('A');
                          ...append 'A'
...'A' appended
```

Expected Result: Server is able to remember the previous commited data "AB" and the current lock holder Client 2, so that Client 2 can continue to send file_append requests after the server recovers from the failure; Client 1 can reaqruie the lock after Client 2 releases the lock; the file contains "ABBA".