Distributed Systems

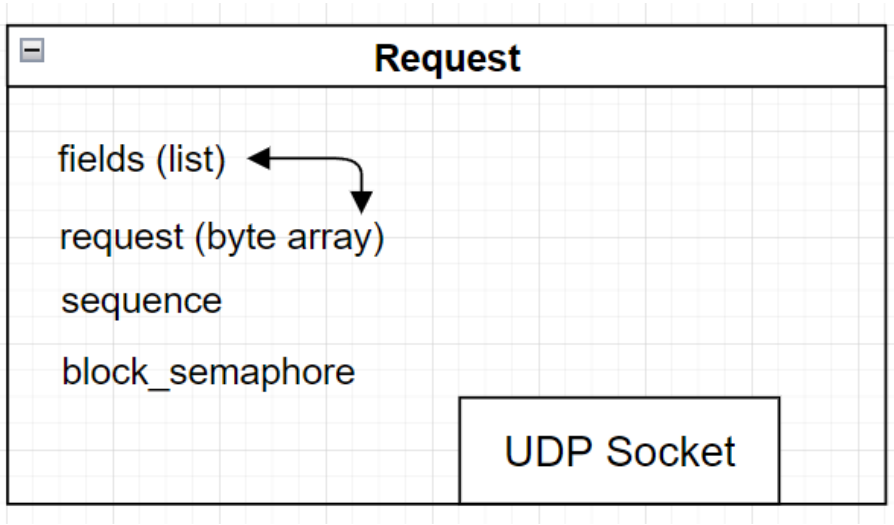
Στολτίδης Αλέξανδρος 2824
Κουτσούκης Νικόλαος 2907

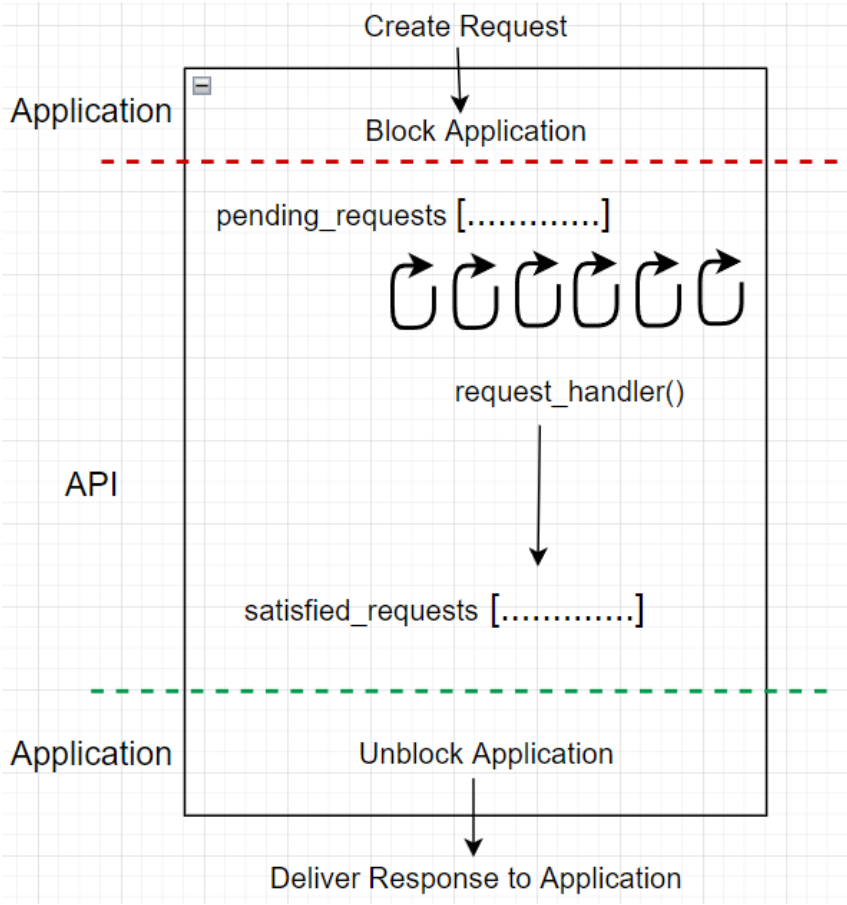
DEPARTMENT OF ELECTRICAL
& COMPUTER ENGINEERING

Client Side

Life of a Request on the Client Side

1. Client Create a Request and Delivers it to the API
2. Client Blocks and Waits for the API to Handle the Request
3. When the Request is Satisfied by the API the Client Continues
4. The Client Finds the Response based on the Request Sequence

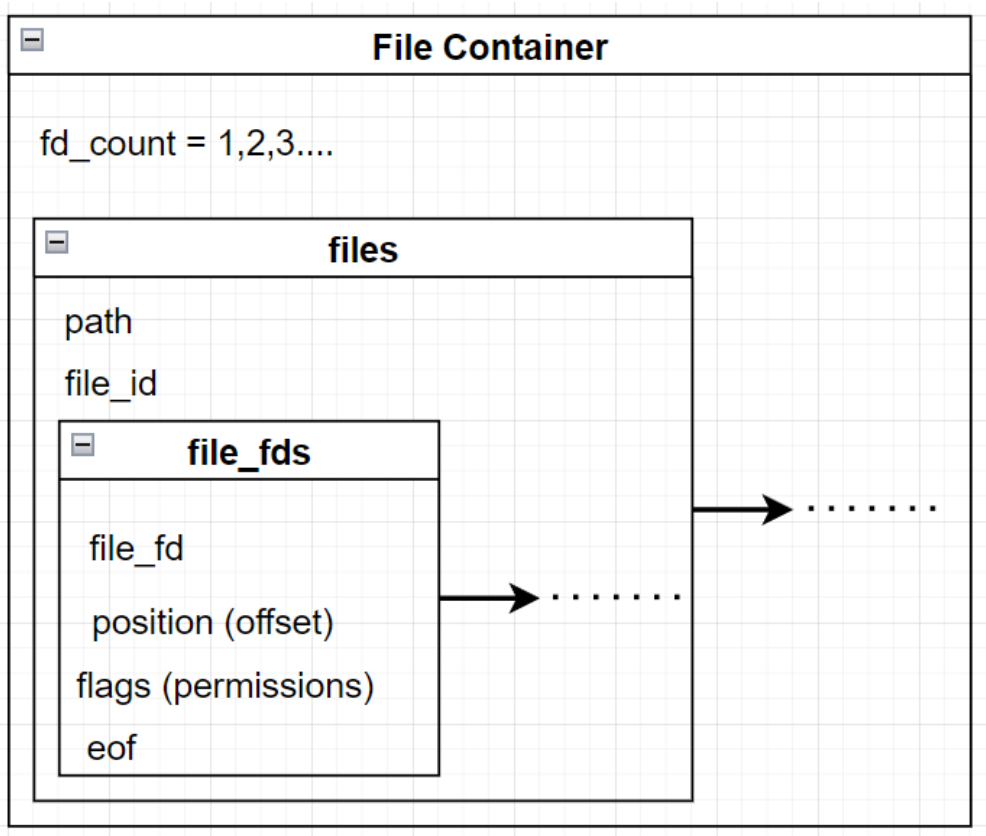




Each Request Contains its own UDP Socket which the API uses to Communicate with the Server

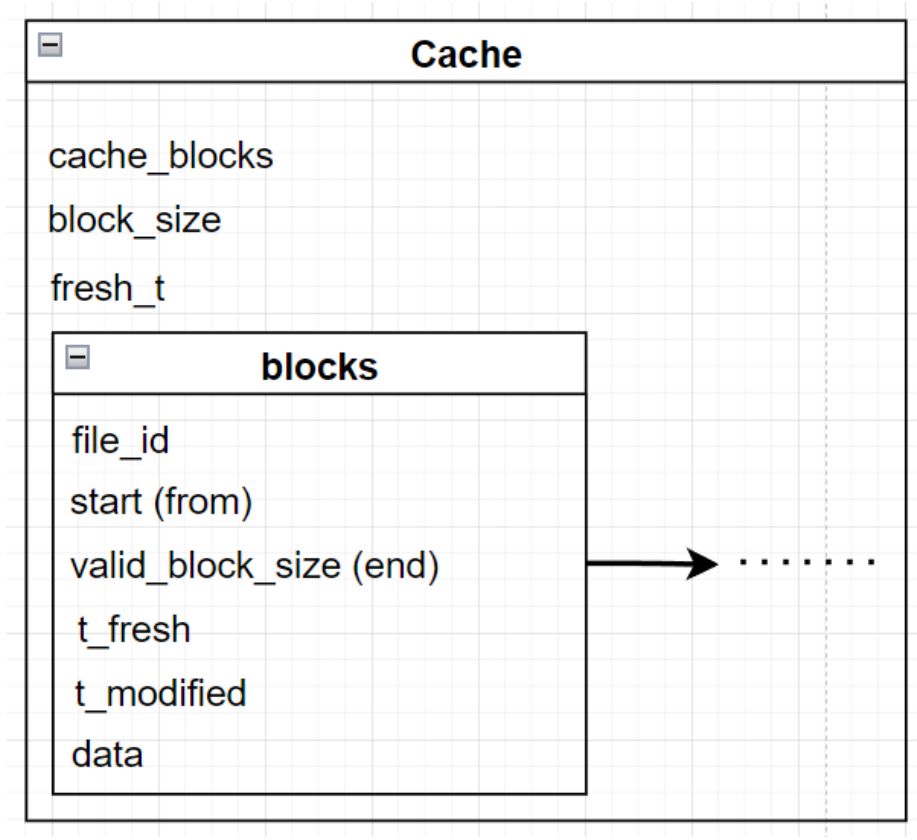
Files on the Client API

1. All Different Files are Stored in a List of Files
2. Each File Contains its File Descriptors, Flags, Offset…
3. When a Client Opens a File a FD is Added
4. One File has Multiple FDs
5. A Single FD Corresponds to a Single File



Cache on the Client API

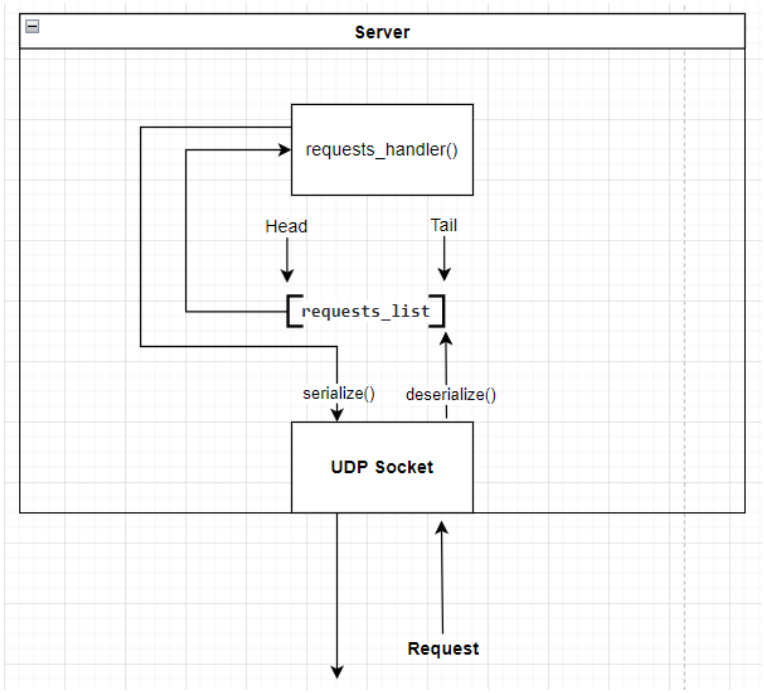
1. A List of Cached Blocks is Stored in Client Memory
2. Each Block Contains its Start Offset and its Size…
3. The Replacement Policy used is the LRU or LRF (LF Fetched)



Server Side

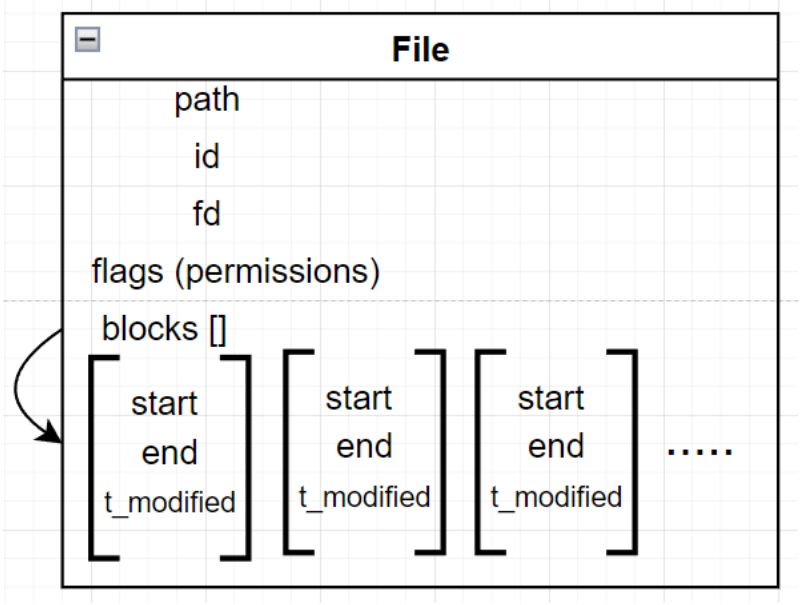
Life of a Request Server Side

1. When a Request is Received it is Deserialized
2. The Request is Appended on a List
3. A thread Checks the List for Available Requests and Pop from the Beginning
4. After the Request is Handled the Response is Serialized and Sent Back to Client



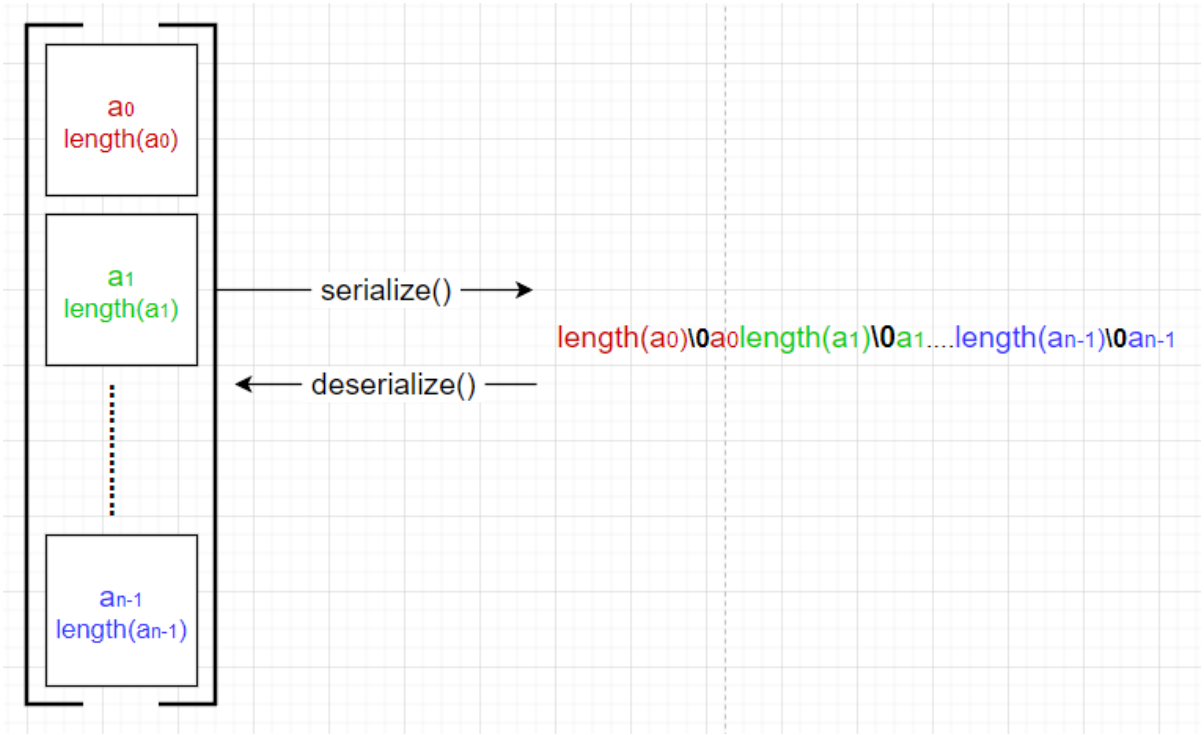
Files and Blocks on the Server Side

1. Each File is Consisted of a Set of Blocks
2. The Start and End of Each Block is Calculated According to the Client's Specified Block
3. A Block Might Overlap with the other Blocks
4. When a Block is Modified Blocks that Overlap are also Modified



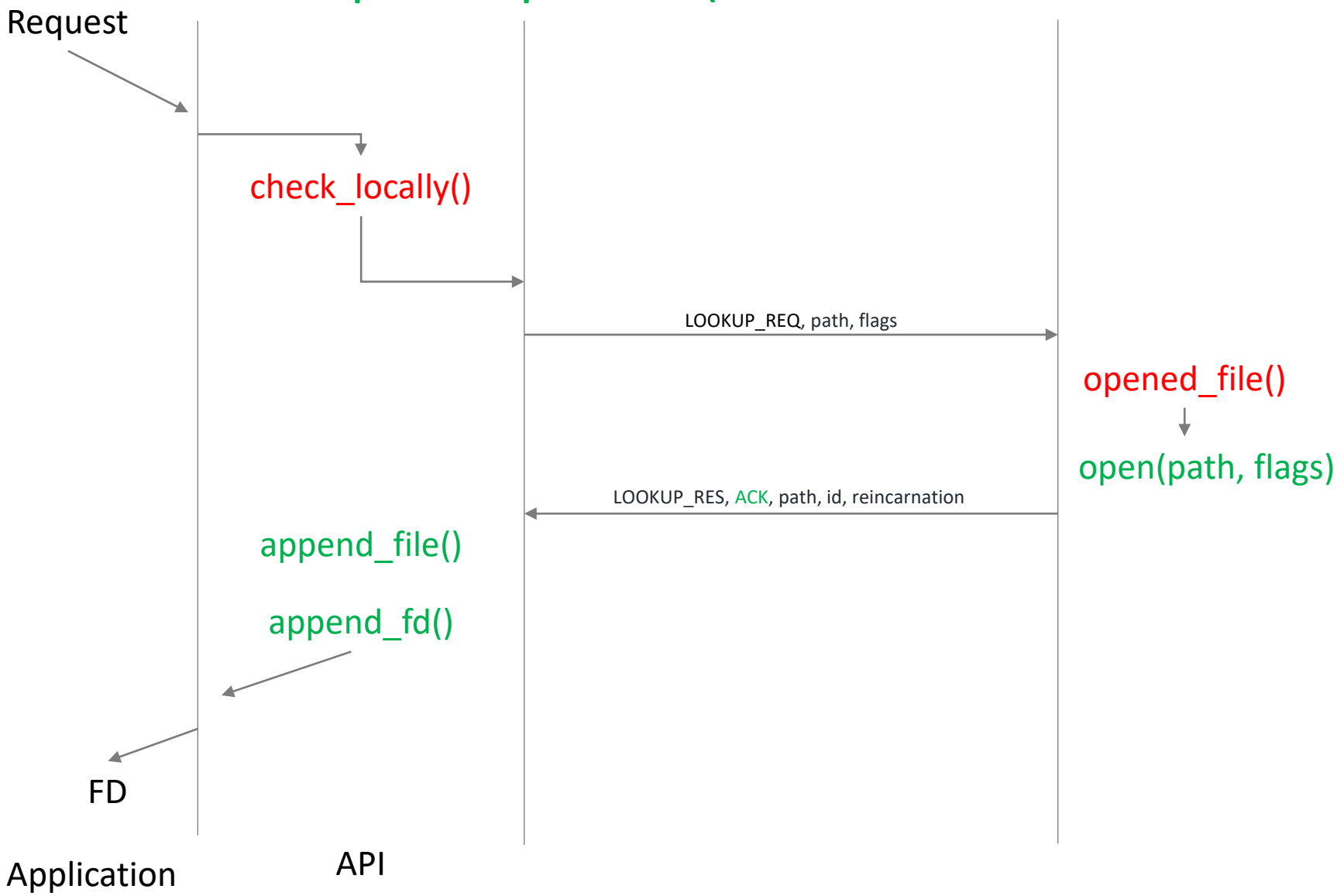
Client - Server Communication

1. Messages are Serialized as Byte Streams and not like Strings
2. Functions on both the Client and the Server Side are implemented to Handle Serialization/Deserialization

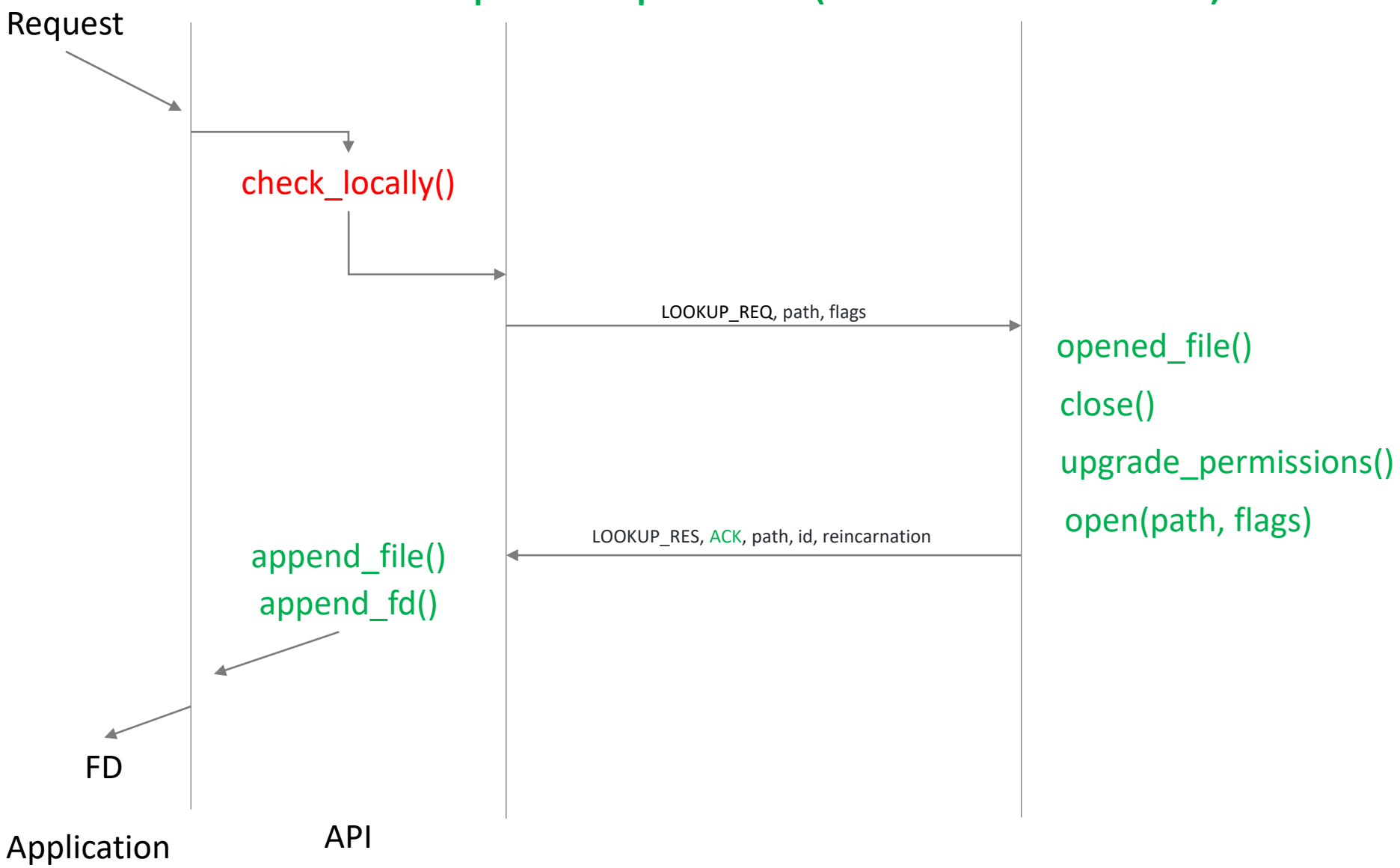


Lookup Requests

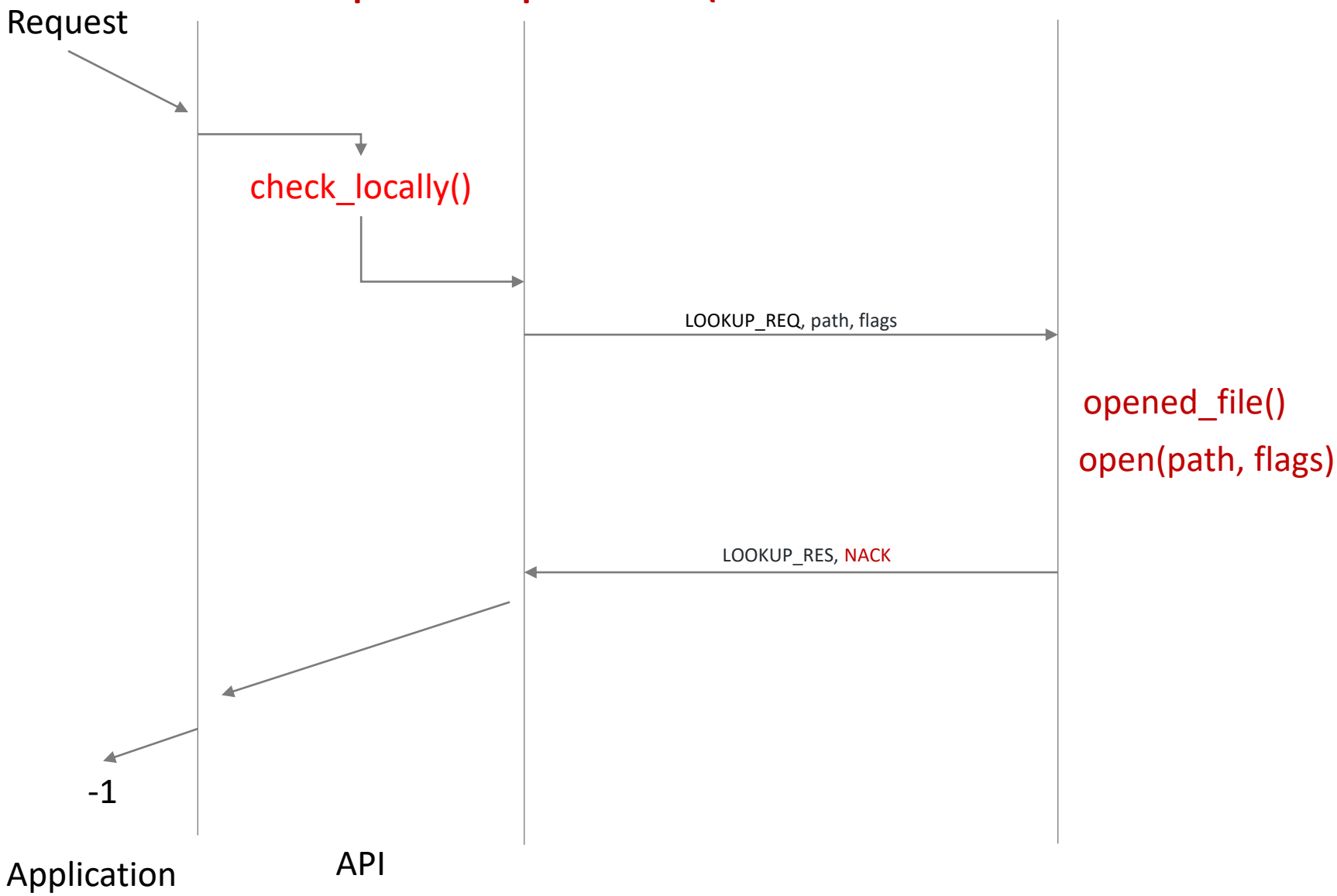
Lookup Request (Fresh Client - Server)



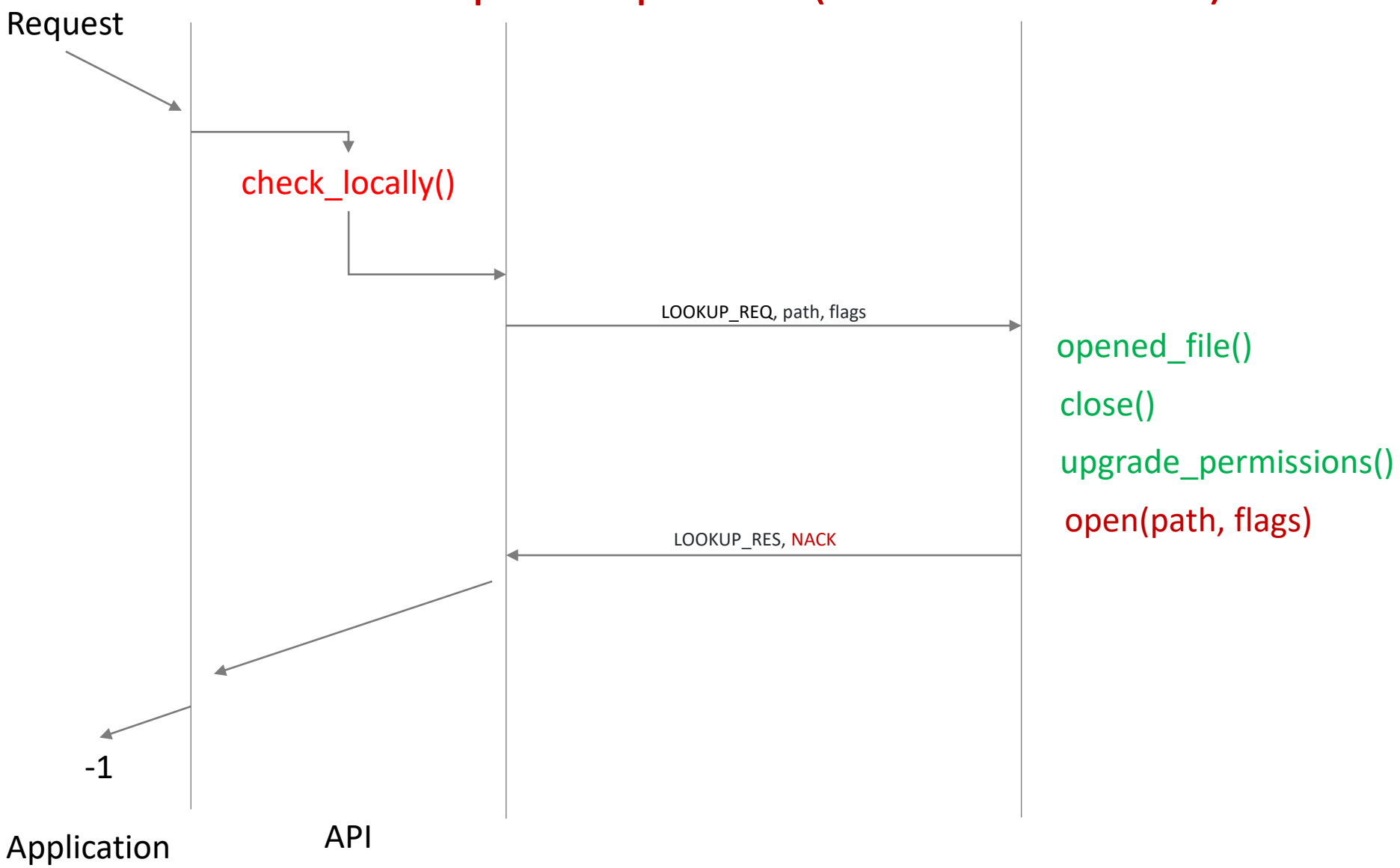
Lookup Request (Fresh Client)



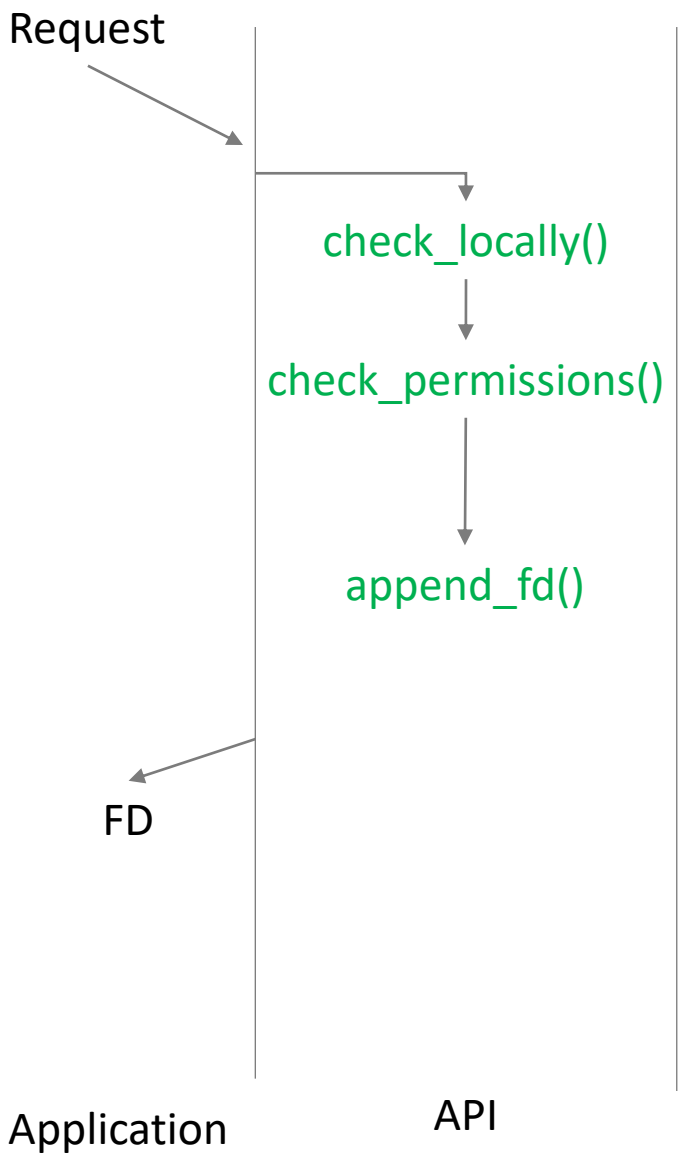
Lookup Request (Fresh Client - Server)



Lookup Request (Fresh Client)

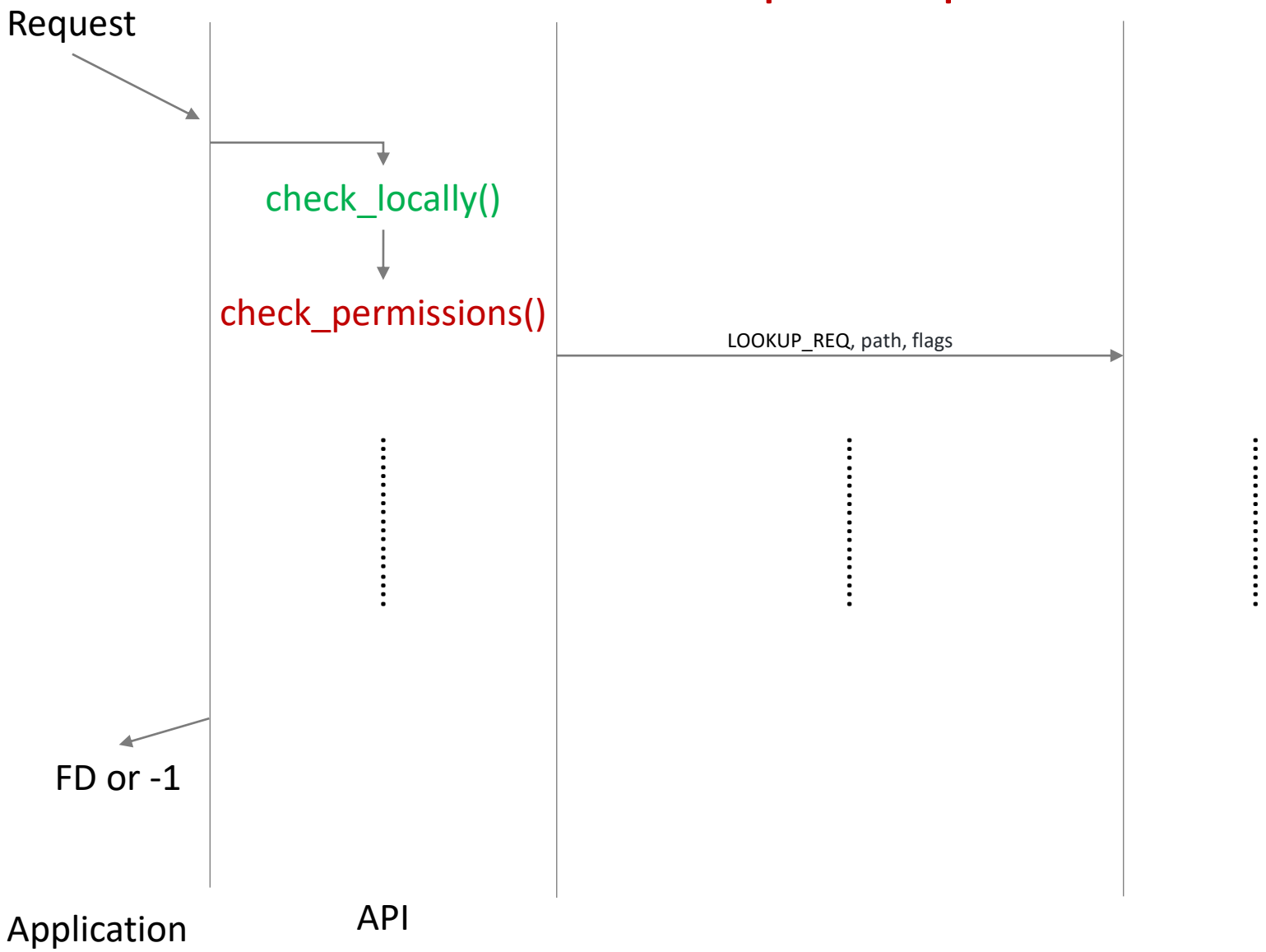


Lookup Request



*check_permissions(): If at least one file contains the requested flags then no request must be send to server to upgrade permissions.

Lookup Request

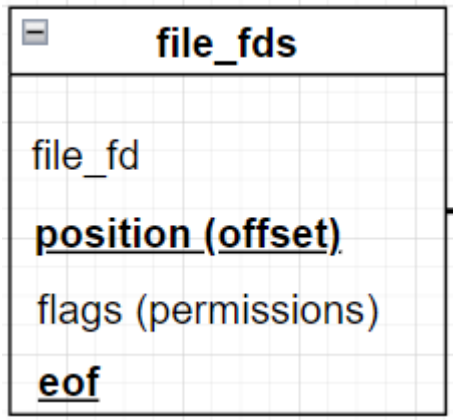


Seek and Close Requests

Seek Request

After the File FD is Found the Position and EOF are Updated as Seen Bellow

Whence	Position	EOF
SEEK_SET	0	False
SEEK_CUR	Current Position	False
SEEK_END	-1	EOF



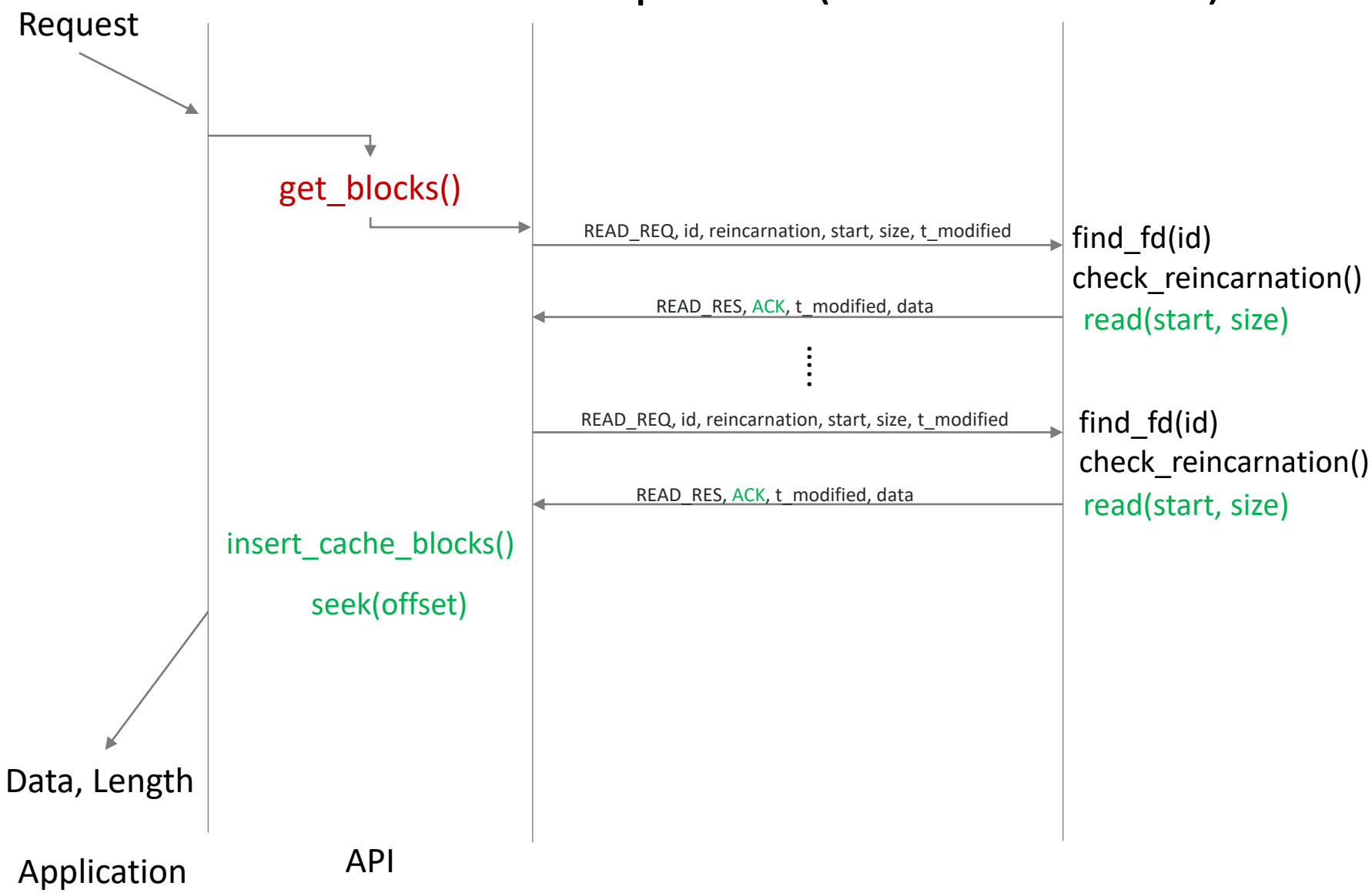
* If Read is Called when the FP is EOF then it Returns Immediately without Request to Server or to Cache

Close Request

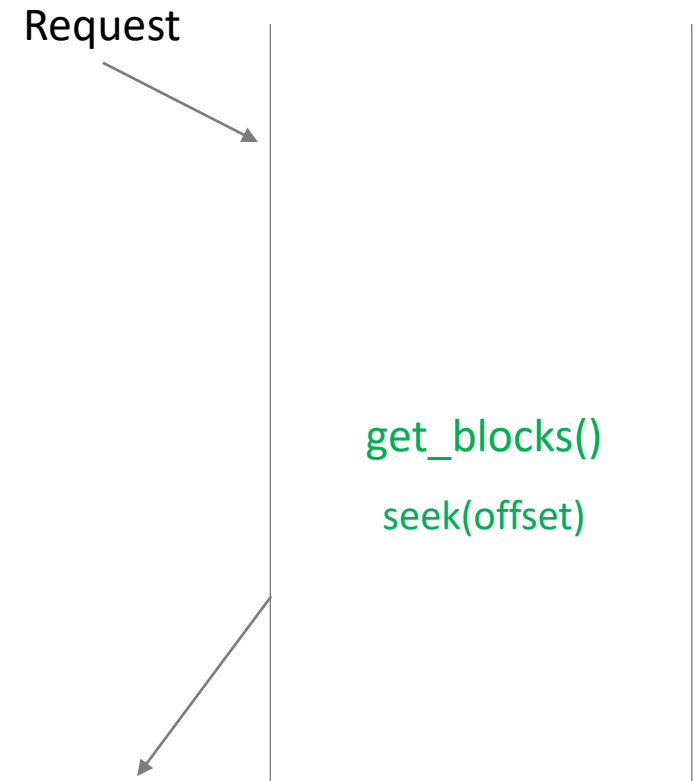
After the File FD is Found it is Removed from the List of all Available FDs

Read Requests

Read Request (Cache Miss)



Read Request (Cache Hit)



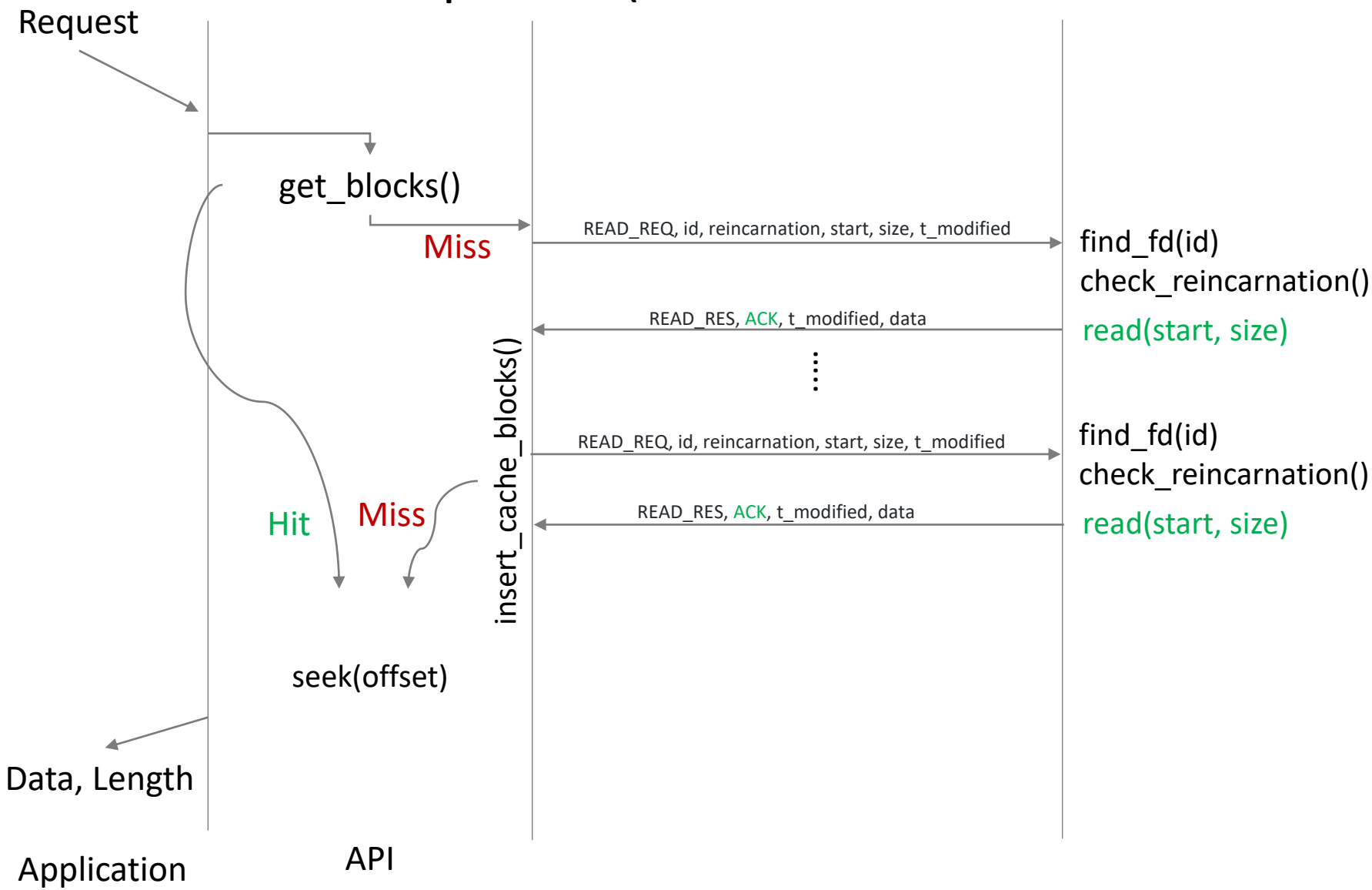
get_blocks()

seek(offset)

Application

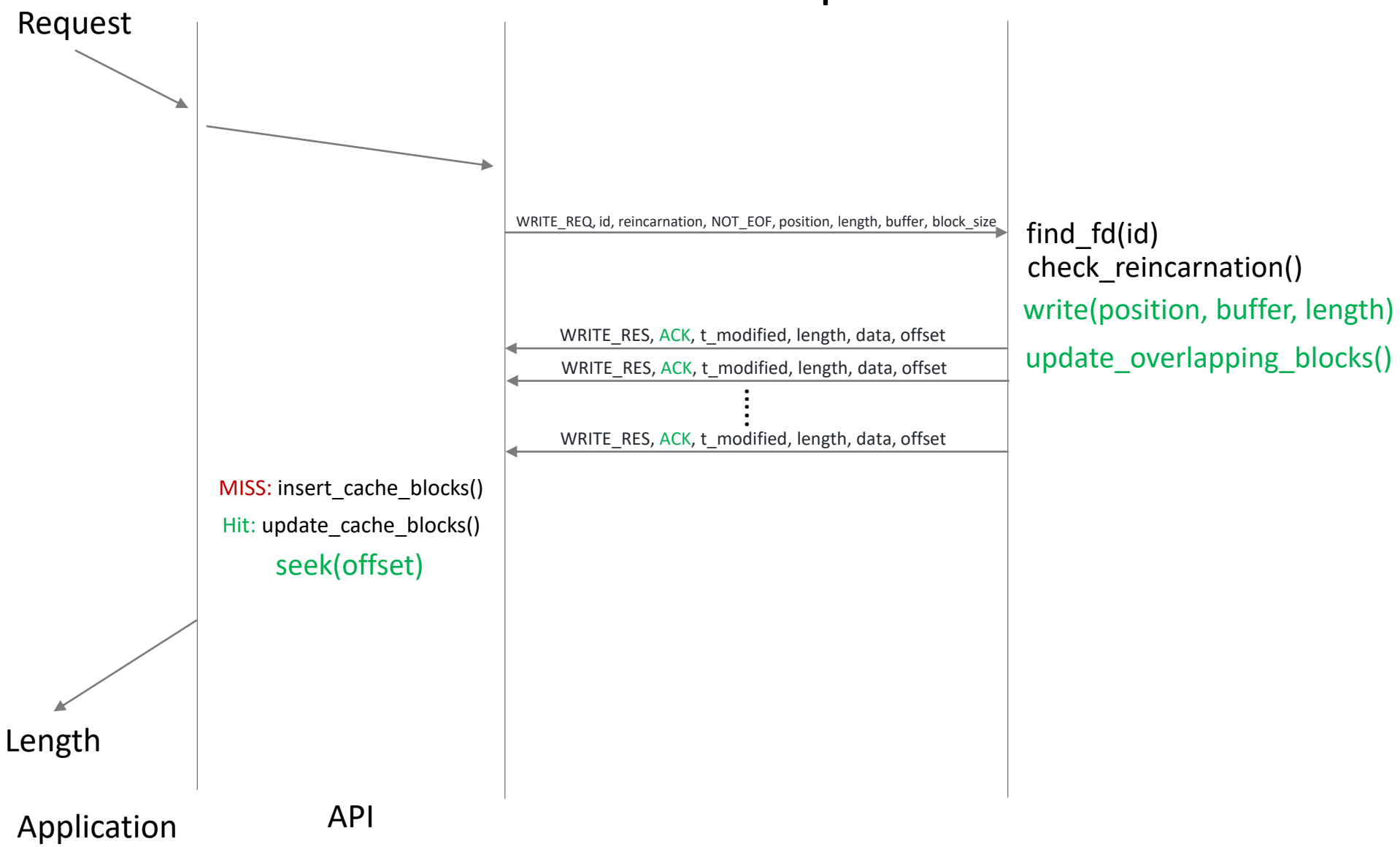
API

Read Request (Partial Cache Hit - Miss)

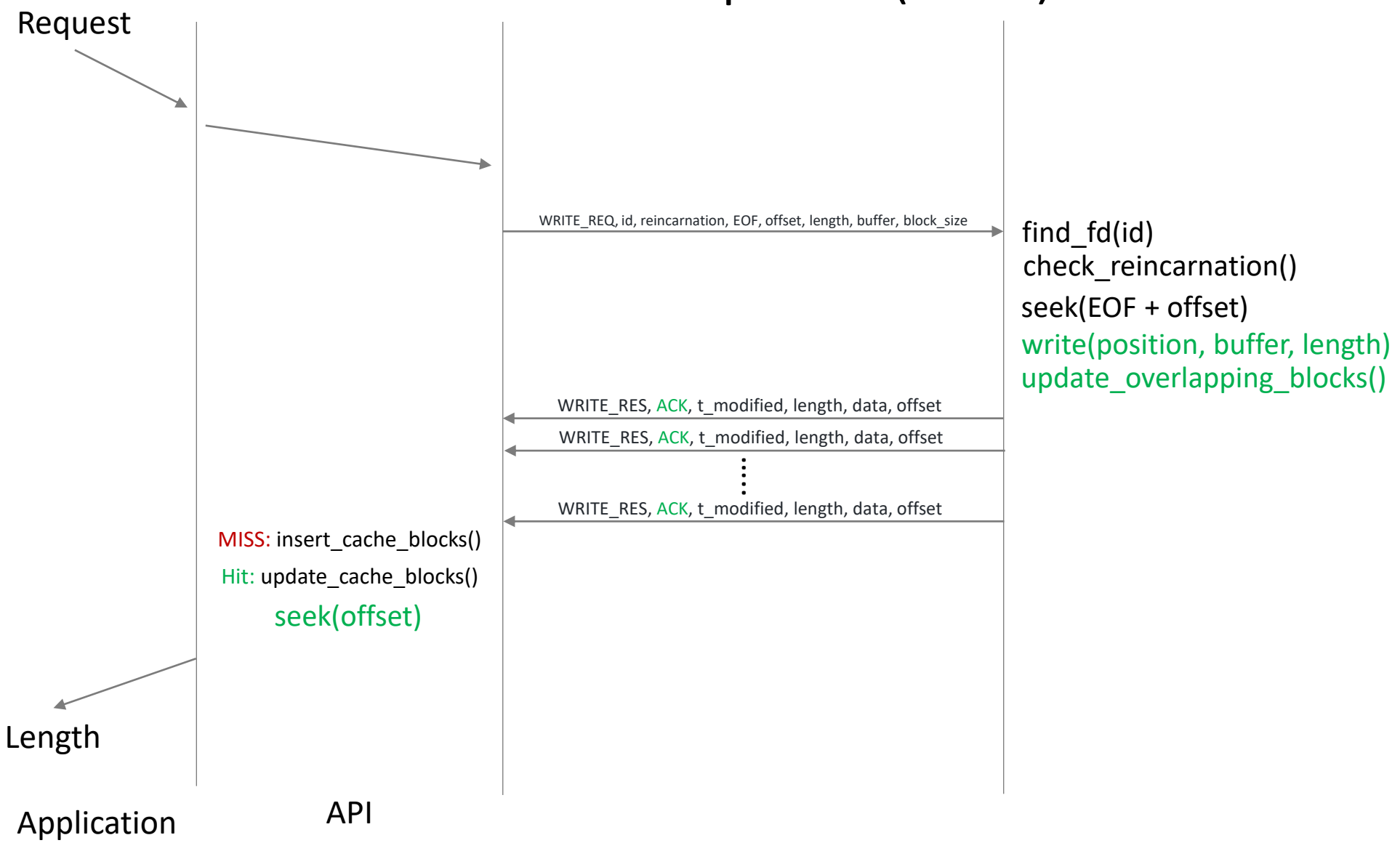


Write Requests

Write Request



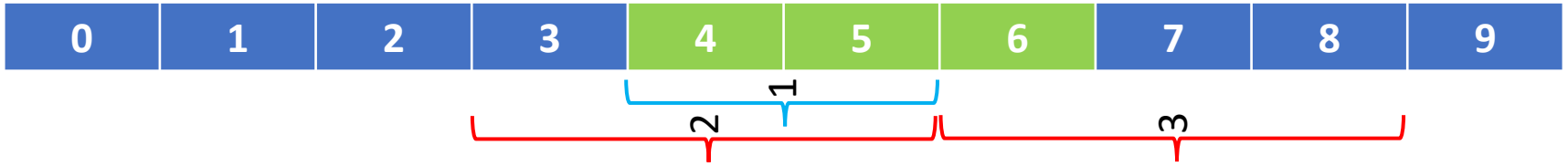
Write Request (EOF)



Dynamic Block Size and Block Overlaps

Simple Example

1. Client (A): Position 4 Bytes, Write 2 Bytes, Block Size 2 Bytes
2. Client (B): Position 4 Bytes, Write 3 Bytes, Block Size 3 Bytes

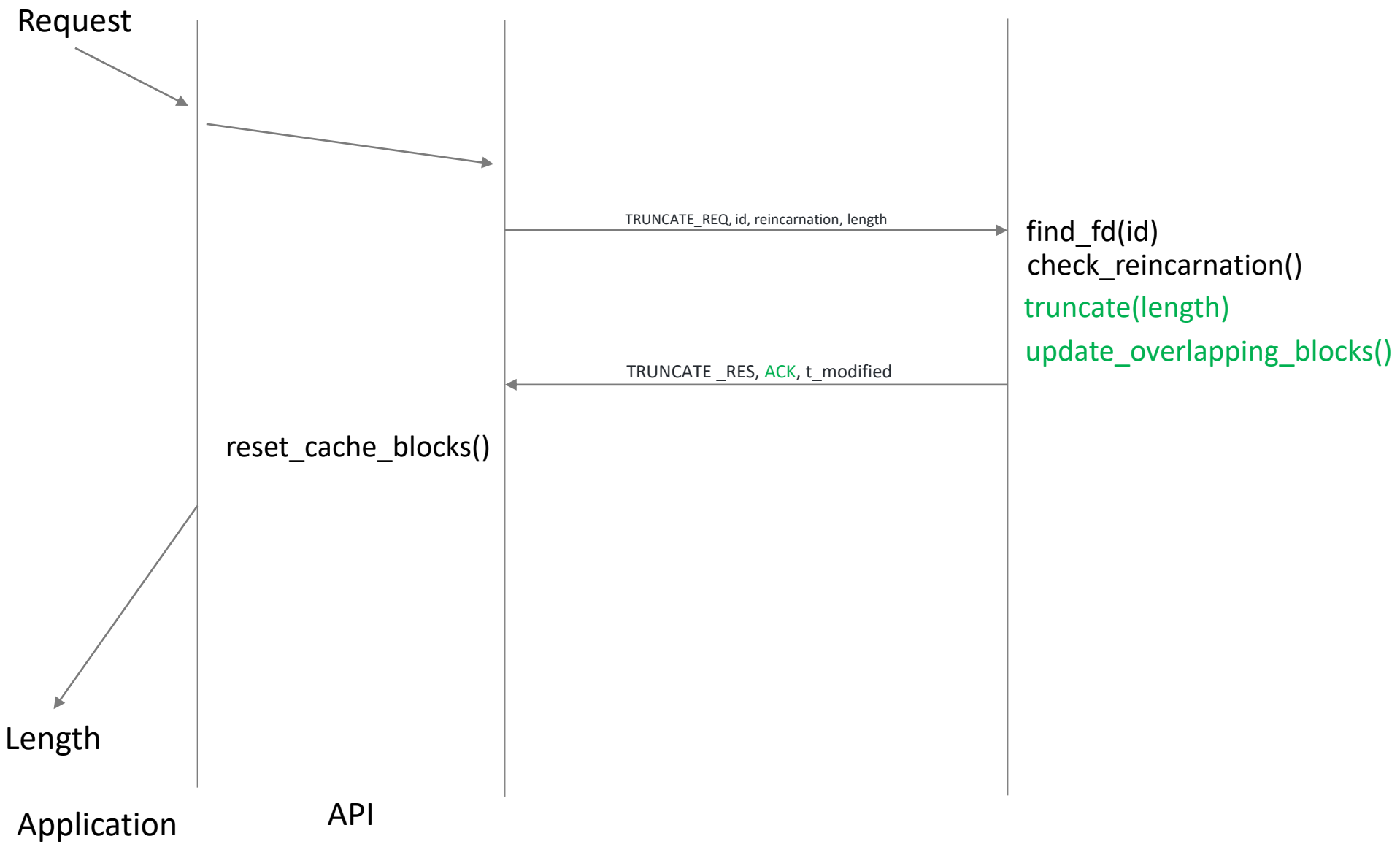


Blocks

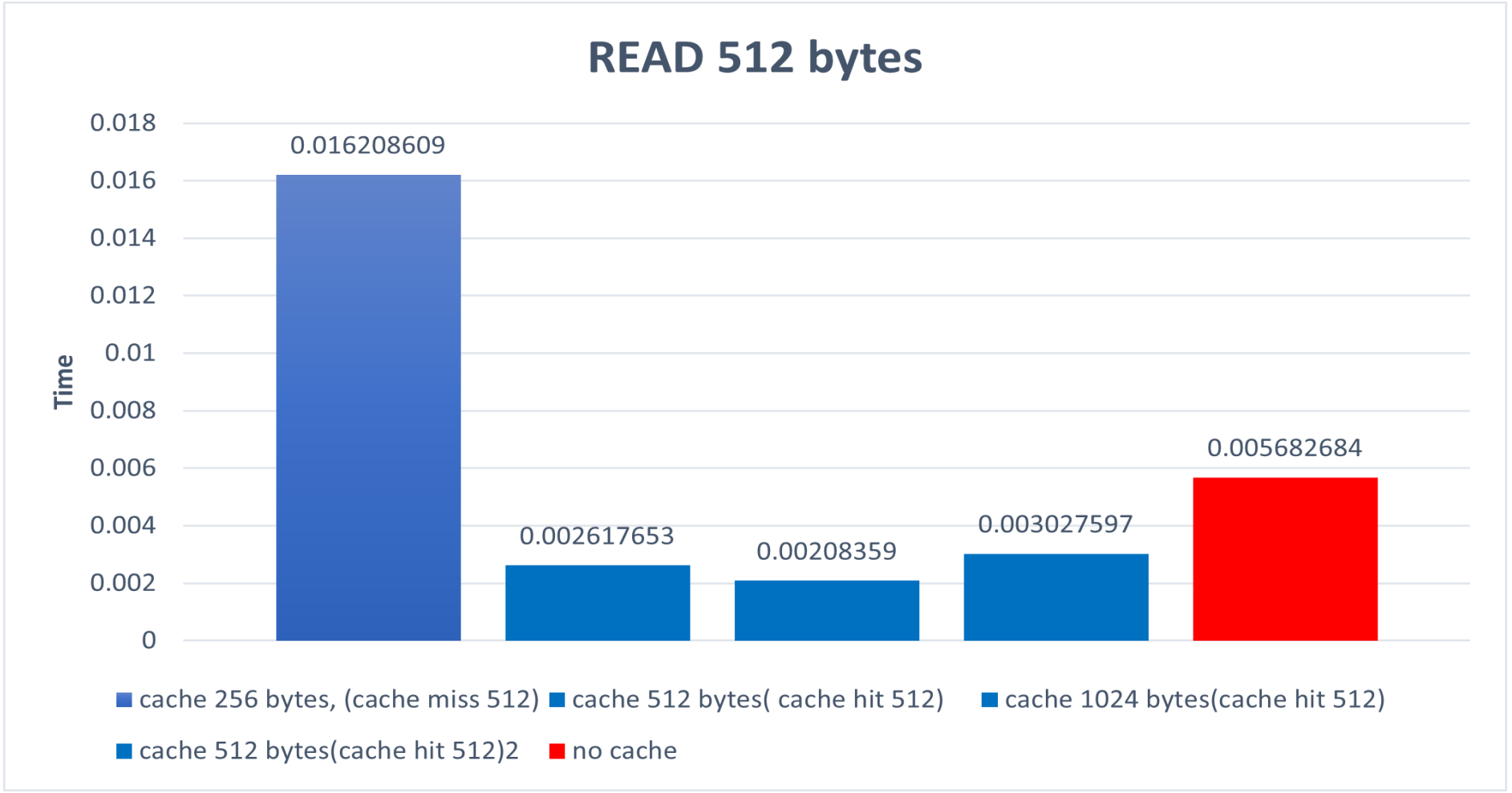
1	2	3
start: 4 end: 6 t_modified: ….	start: 3 end: 6 t_modified: ….	start: 6 end: 9 t_modified: ….

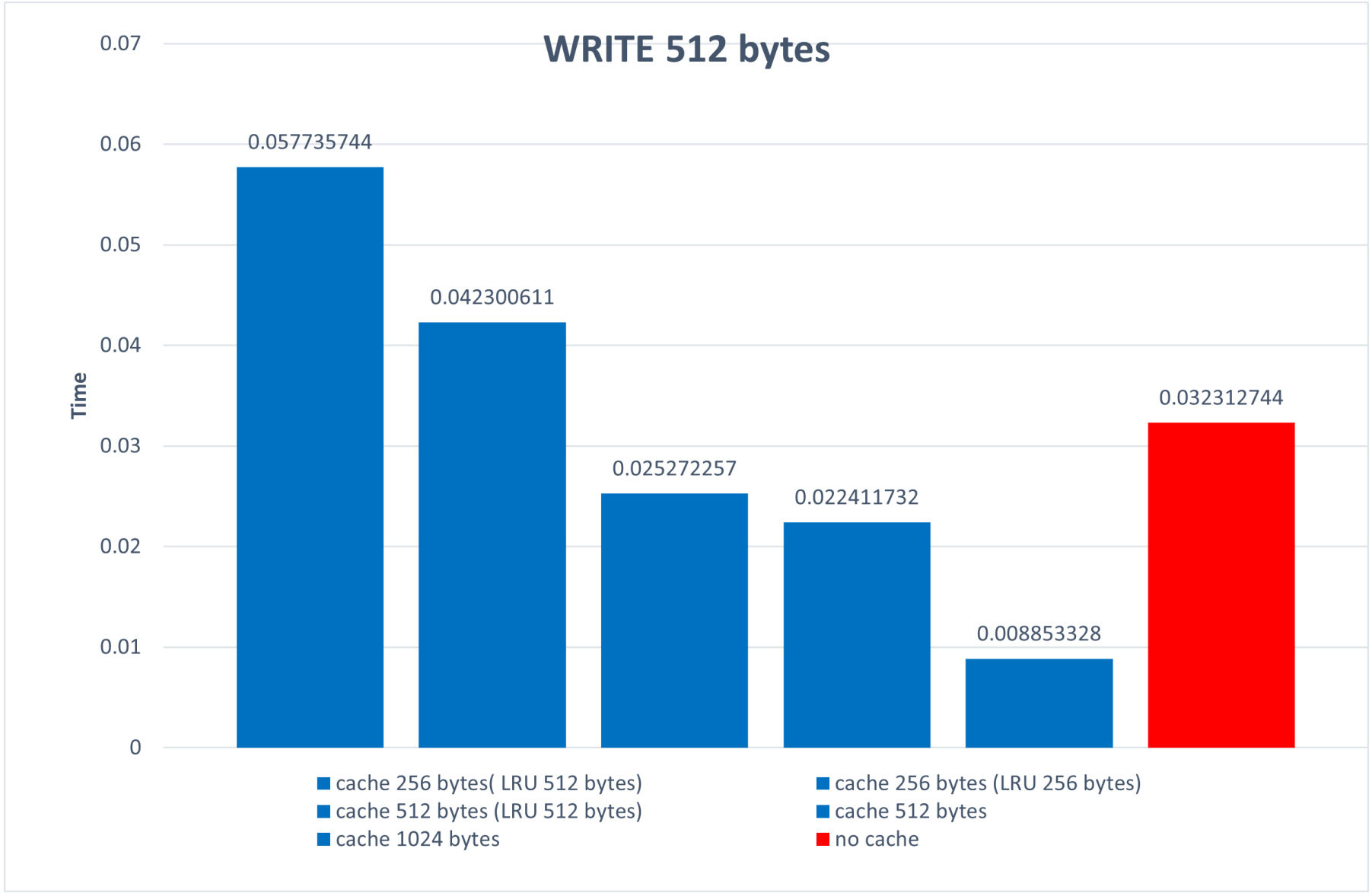
* Since an Overlap Exists the t_modified of block 1 must also be updated.

Truncate


Request
TRUNCATE_REQ, id, reincarnation, length
find_fd(id)
check_reincarnation()
truncate(length)
update_overlapping_blocks()
TRUNCATE _RES, ACK, t_modified
reset_cache_blocks()
Length
Application
API

Benchmarks


READ 512 bytes
Time
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0.016208609
0.002617653
0.00208359
0.003027597
0.005682684
cache 256 bytes, (cache miss 512)
cache 512 bytes(cache hit 512)
cache 1024 bytes(cache hit 512)
cache 512 bytes(cache hit 512)2
no cache


WRITE 512 bytes
Time
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0.057735744
0.042300611
0.025272257
0.022411732
0.008853328
0.032312744
cache 256 bytes(LRU 512 bytes)
cache 512 bytes (LRU 512 bytes)
cache 1024 bytes
cache 256 bytes (LRU 256 bytes)
cache 512 bytes
no cache

Group Members

- Αλέξανδρος Στολτίδης 2824 (stalexandros@uth.gr)
- Νικόλαος Κουτσούκης 2907 (nkoutsoukis@uth.gr)