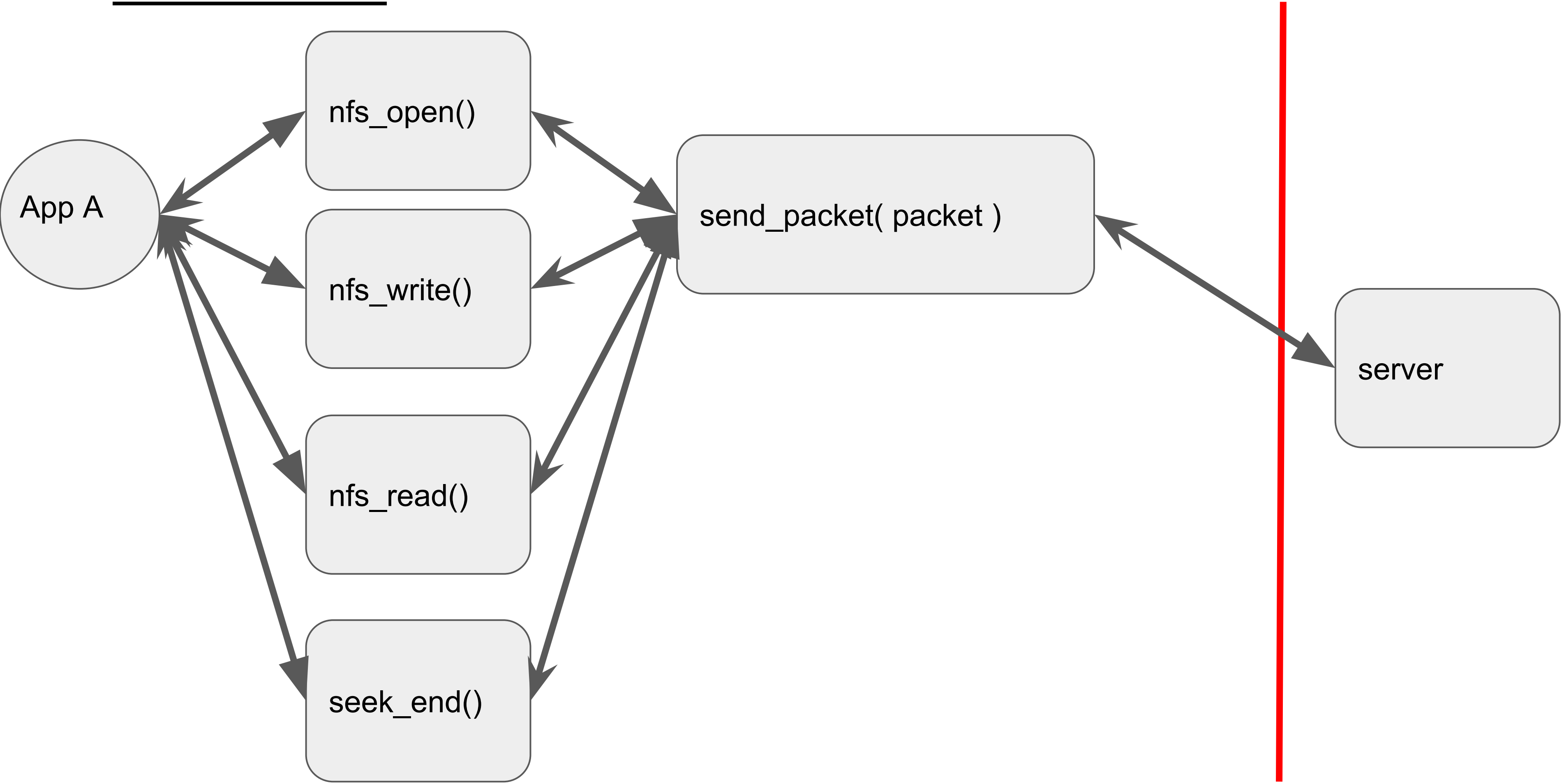
Κατανεμημένα συστήματα

3η εργασία: Ομαδική επικοινωνία
(Ημ.παράδοσης:5/5/2018)

Ράντου Καλλιόπη(2004)
Νικητάκης Παναγιώτης(1717)

NFS - Protocol



App A

nfs_open()

nfs_write()

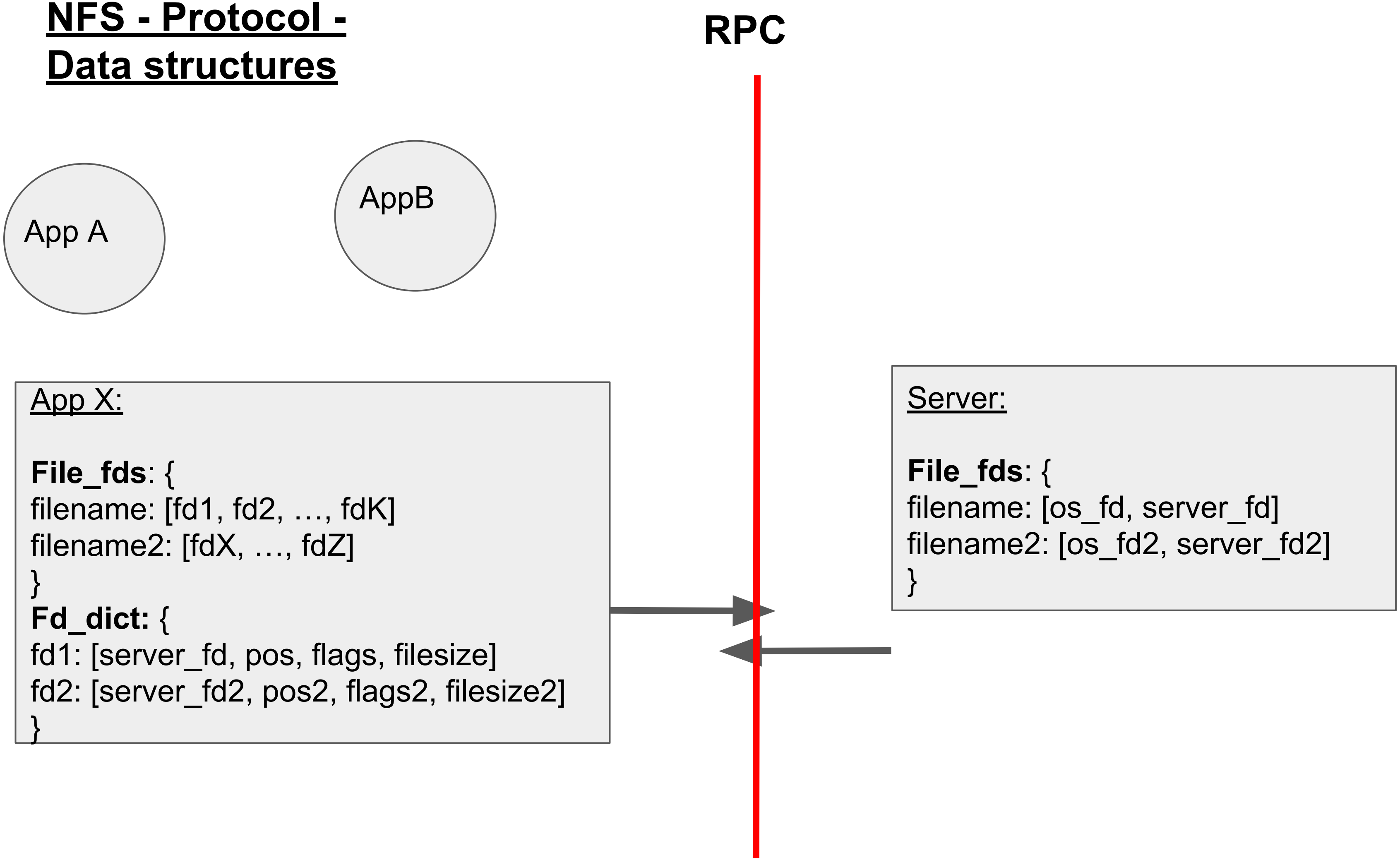
nfs_read()

seek_end()

send_packet(packet)

server

NFS - Protocol - Data structures



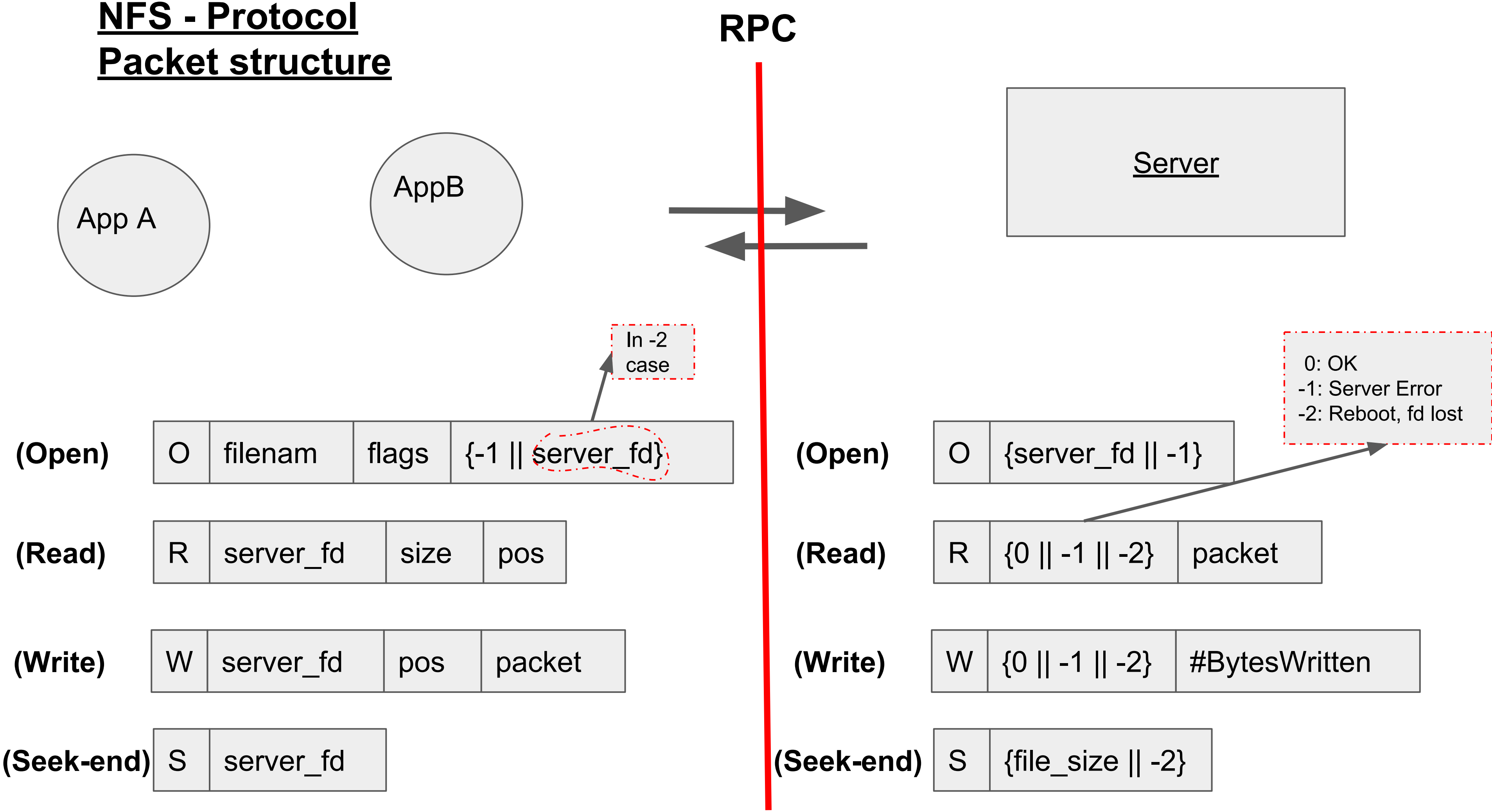
App X:

File_fds: {
filename: [fd1, fd2, …, fdK]
filename2: [fdX, …, fdZ]
}
Fd_dict: {
fd1: [server_fd, pos, flags, filesize]
fd2: [server_fd2, pos2, flags2, filesize2]
}

Server:

File_fds: {
filename: [os_fd, server_fd]
filename2: [os_fd2, server_fd2]
}

NFS - Protocol Packet structure



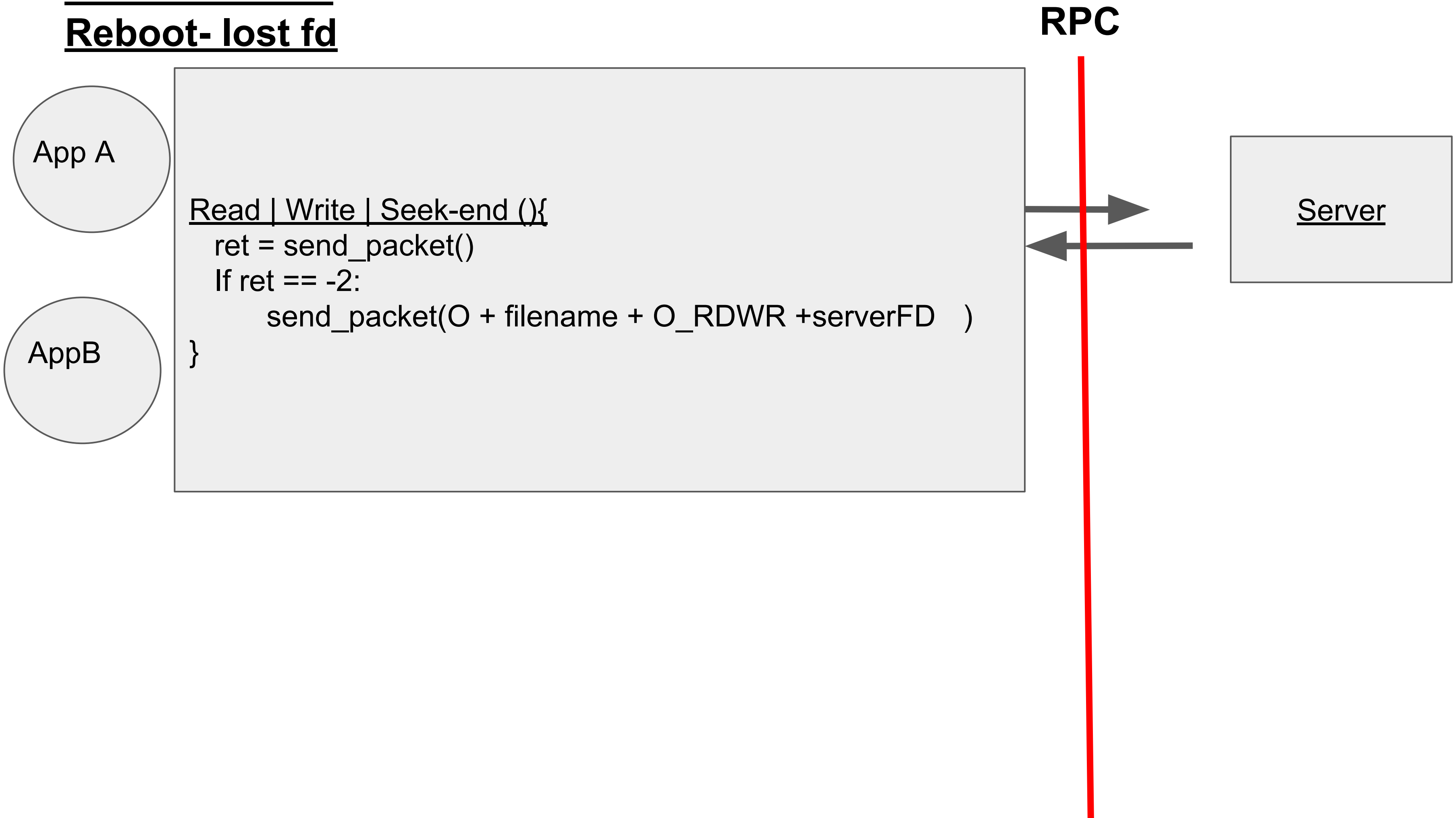
In -2 case

0: OK
-1: Server Error
-2: Reboot, fd lost

(Open)	O	filenam	flags	{-1 || server_fd}
(Read)	R	server_fd	size	pos
(Write)	W	server_fd	pos	packet
(Seek-end)	S	server_fd		

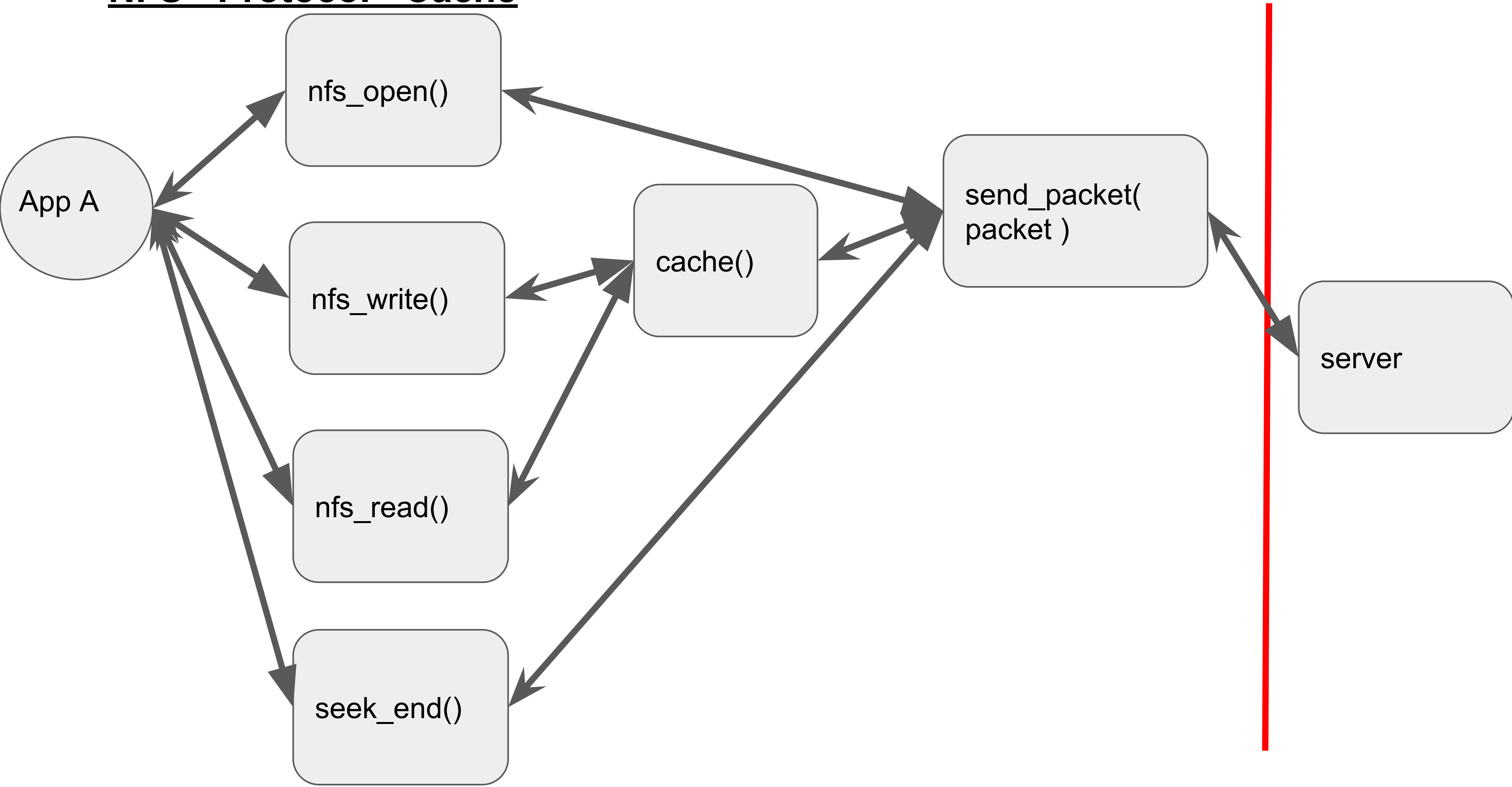
(Open)	O	{server_fd || -1}	
(Read)	R	{0 || -1 || -2}	packet
(Write)	W	{0 || -1 || -2}	#BytesWritten
(Seek-end)	S	{file_size || -2}	

NFS - Protocol
Reboot- lost fd



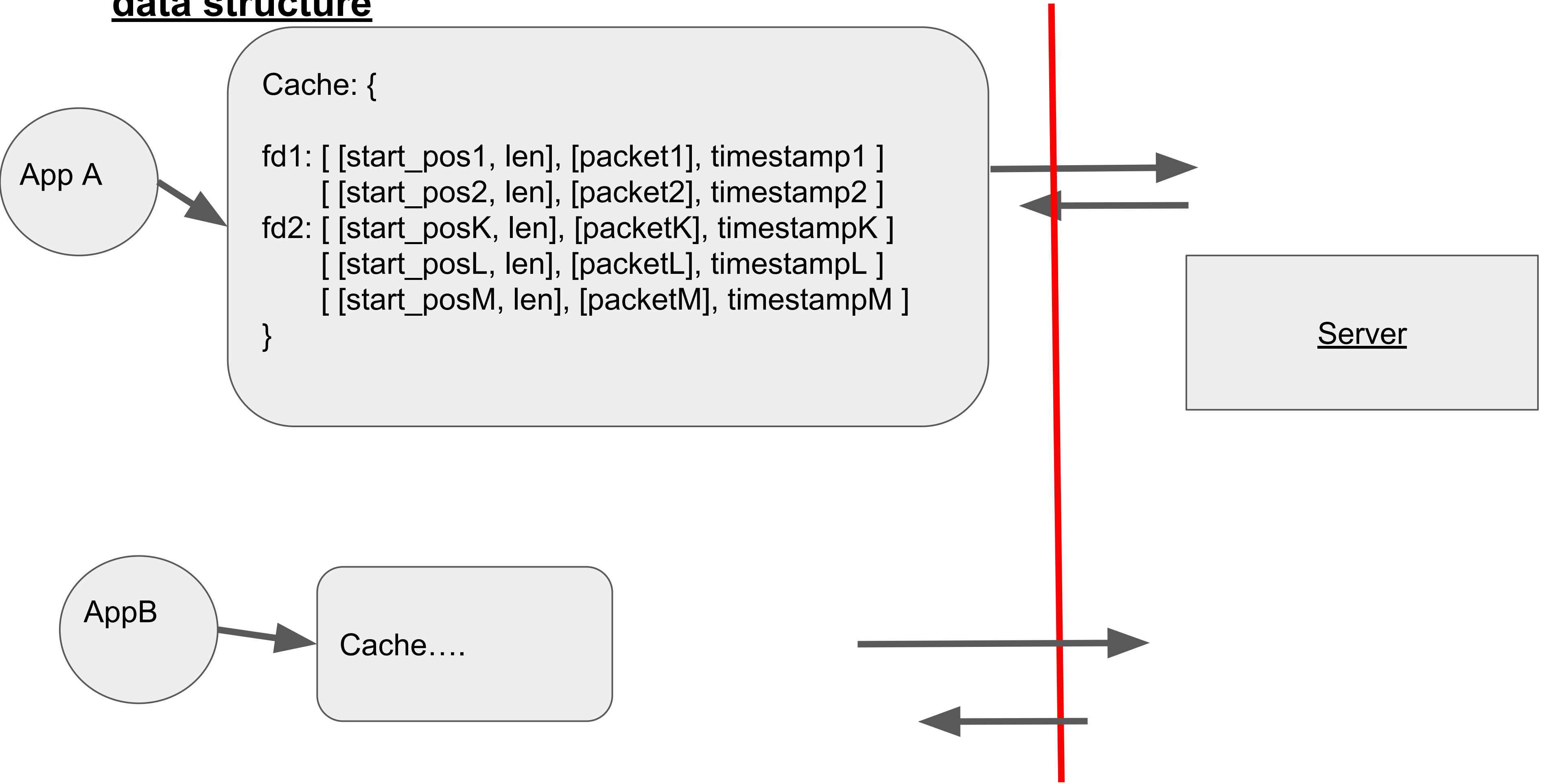
```
Read | Write | Seek-end (){
   ret = send_packet()
   If ret == -2:
         send_packet(O + filename + O_RDWR +serverFD    )
}
```

NFS - Protocol - Cache



NFS - Protocol - Cache data structure

RPC



App A

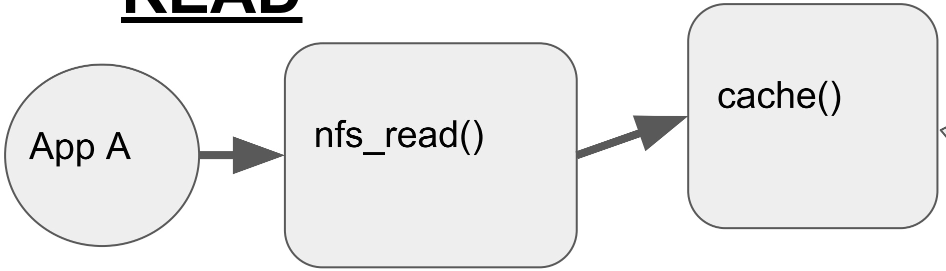
```
Cache: {

fd1: [ [start_pos1, len], [packet1], timestamp1 ]
       [ [start_pos2, len], [packet2], timestamp2 ]
fd2: [ [start_posK, len], [packetK], timestampK ]
       [ [start_posL, len], [packetL], timestampL ]
       [ [start_posM, len], [packetM], timestampM ]
}
```

Server

AppB

Cache….

NFS - Implementation - READ

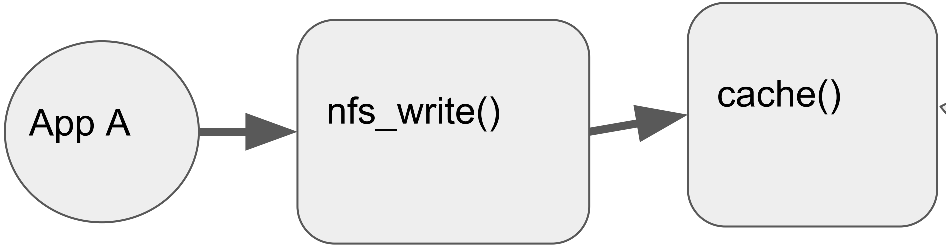


```
blocks = file_size / cache_block_size
while file_size > 0:
 If server_fd in cache:
  for each line:
   If file_pos in line:
    buffer.append(line)
   else:
      Find block_start
      ret= send_packet(‘R’,server_fd,
block_size, bloc_start)
       ret=data
       Ret = read_cache(data)
       If ret == 1: #end offile
         buf.append(reast_packet)
  Else:
   Find block_start
      ret= send_packet(‘R’,server_fd,
block_size, bloc_start)
       ret=data
       Ret = read_cache(data)
```

read_cache()

```
total_blocks=0
for x in cache_dic.keys():
   # Find total_blocks
if total_blocks > configuration.cacheSize:
   minLRU = time.time()
   Find least recem used block
   Delete block
if total_blocks <= configuration.cacheSize
  if server_fd in cache_dic::
    cache_dic[server_fd].insert(sortedpos, data)
else:   #an den exw fd in cache
  cache_dic[server_fd]=[[[file_pos, len(data)],[data],
time.time()]]
```

NFS - Implementation - WRITE



```
 ret= send_packet()
If ret >0:
  blocks = file_size / cache_block_size
  while temp_len > 0:
  If server_fd in cache:
    if len(cache_dic[server_fd]) == 0:
        write_cache()
    for each line:
        If file_pos in line:
          Find block start
          Create block #me bash block_start kai file_pos
          write_cache(server_fd, temp_block, block_start, configuration.blockSize)
        Else:#an den uparxei o filepos
          Find block start
          cache_dic[server_fd].insert(sorted_pos, tempblock)
  Else:#an den uparxei o fd
    write_cache(server_fd)
```

Write_cache()

```
total_blocks=0
for x in cache_dic.keys():
   # Find total_blocks
if total_blocks > configuration.cacheSize:
   minLRU = time.time()
   Find least recem used block
   Delete block
if total_blocks <= configuration.cacheSize
if server_fd in cache_dic::
  for x in cache_dic[server_fd]:
   If file_pos:
    #ananewsh thesewn sthn cache_dic
else:   #an den exw fd in cache
  cache_dic[server_fd]=[]
```