

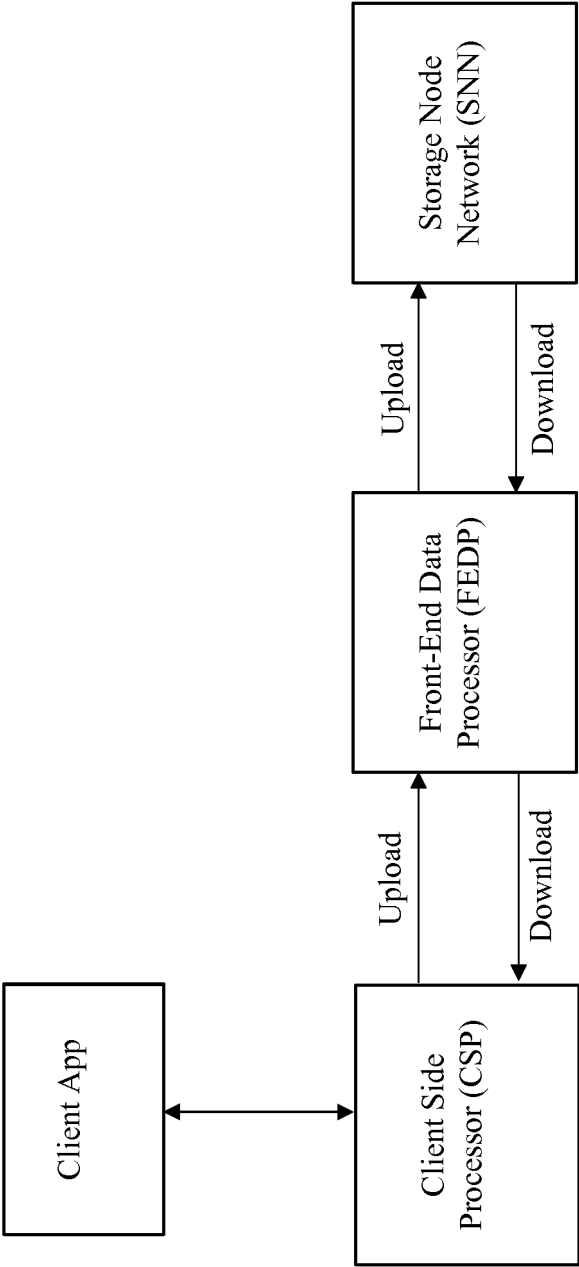


FIG. 1

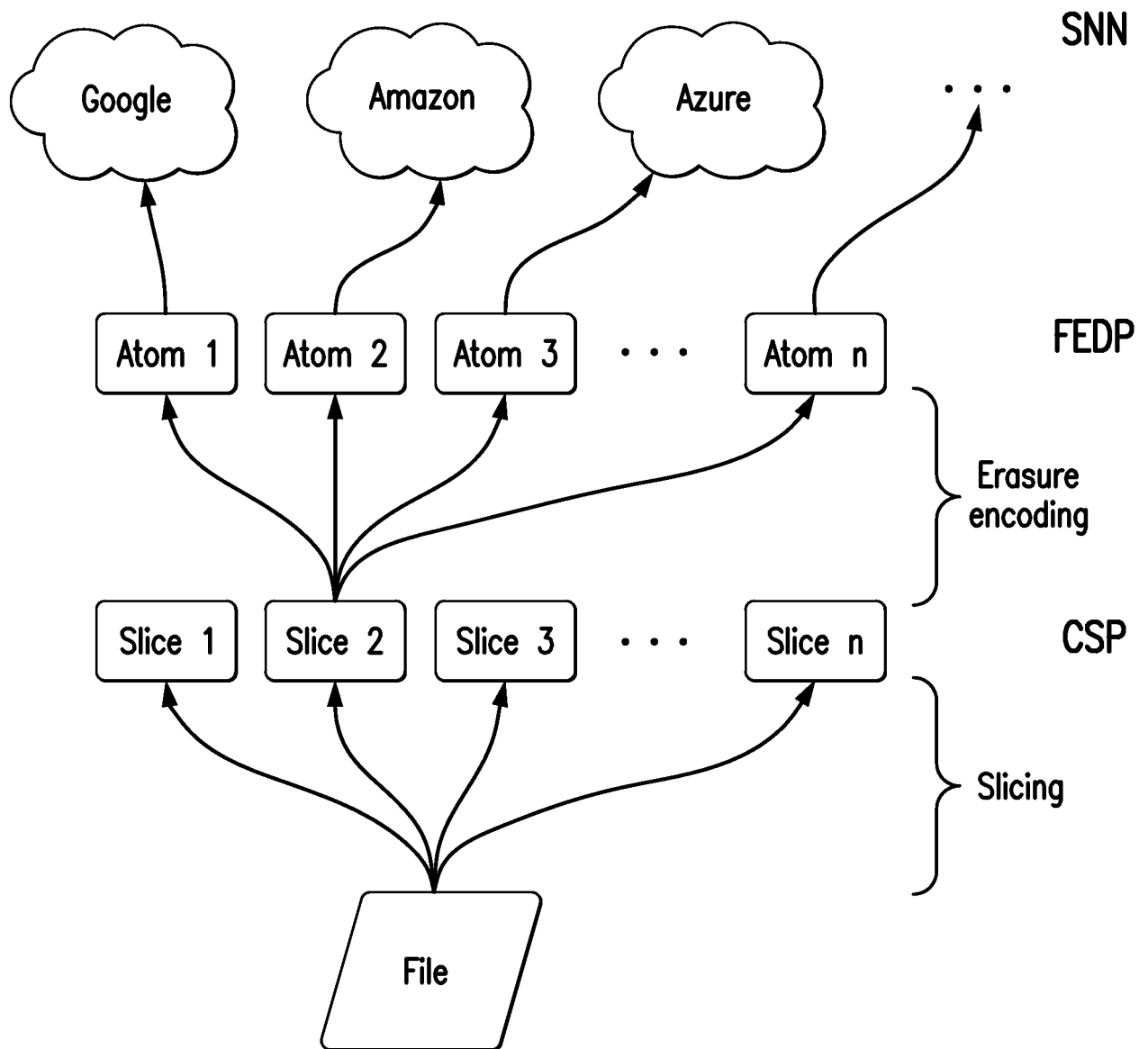


FIG. 2

1	Client app sends requests to upload file to CSP for file in client datacenter. The client app also sends to the CSP the client's list of optimized SNN's for use later in storing the file fragments. A metadata file <i>metadata0</i> is created to describe the files physical parameters in the client's datacenter (size, location, checksum, etc.). This metadata file is encrypted and stored in the client's datacenter and also sent to storage nodes.	Whole file in data center	Client app
2	CSP assigns unique file ID (FID) to the file about to be upload	Whole file in data center	CSP
3	CSP slices file into n slices each of size x . The number of slices n and the size x , based on the size of the whole file is a parameter which can be configured by the client app, and is passed to the CSP. A typical value for the slicing: a 1Gb file is sliced into 500 slices. Presently, the slices are of equal size, although a future enhancement will see more intelligent slicing based on a more complicated algorithm to optimize client security. Each slice contains its raw data and the information on client parameters on which SNN's to use for storing this file, as transmitted previously by the client app.	n slices of size X	CSP
4	Each slice is assigned a unique slice ID (SID) which is used to map the slices for reassembly	n slices of size x	CSP
5	A second metadata file <i>metadata1</i> is created which maps how these slices with their unique slice ID (SID) are reassembled to the original file. This metadata file is encrypted using the client's encryption key, and stored at the client's datacenter, it is sent into storage nodes as well.	<i>metadata1, encrypted</i> , maps slices	CSP
6	Each slice is encrypted using the client's encryption key. Only the client can read the slice. Symmetric encryption is used.	n slices of size x encrypted	CSP
7	The CSP sends the encrypted slices on the FEDP's. Latency optimization is performed at the CSP in choosing which FEDP's to use for processing the slices. This is necessary as the client's data centers may lack optimal speed and availability in regards to the FEDP's. Once chosen, the slices are sent out randomly to FEDP's for the next stage of processing.	n slices of size x encrypted	CSP
8	FEDP's are data processing servers maintained by the inventions' servers for the next stage of processing, with highly optimized performance, latency and availability to the SNN's. Optimally, there should be twice as many FEDP's as SNN's in a given client setup to optimize the speed and security improvements in the invention's technology. Normally, these FEDP's will be located in the cloud, outside the client's data center, although the client could also specify that FEDP's would also be at the client's own site to optimize performance. The number of FEDP servers used is optimized for the size of the files transferred, and the available bandwidth.	n slices of size x encrypted, normally outside client's site.	FEDP

FIG. 3

9	Each FEDP uses Reed-Solomon Erasure coding to split up each slice into slice fragments adding some data used in the Erasure coding error detection and correction schemes. A typical example from above, our 1 Gb. file was split initially (at the CSP) into 500 equal size slices, and each slice is split at an FEDP into 15 slice fragments using Erasure coding. Erasure coding is a very CPU intensive process, and these servers have been built with optimum performance for this processing.	a slice fragments per slice created for each slice using Erasure coding.	FEDP
10	Each slice fragment is given an "Fragment ID", and assigned to one of the previously designated SNN data centers. A metadata file <i>metadata2</i> is created which maps each of the slice fragments to the slice from which it was generated, and on which SNN it is to be stored. This metadata file is encrypted using the server's encryption key, and sent to each of the SNN's used in storing the slice fragments which make up the slices, which make up the original file. Thus, each SNN contains a redundant copy of <i>metadata2</i> .	Metadata file <i>metadata2</i> is created to map all the slice fragments to their original slices, and sent to each SNN.	FEDP
11	Each of the slice fragments is encrypted using the server's encryption key, and sent to the SNN designated for those slice fragments. There is no need for latency optimization between the FEDP's and SNN's, as FEDP's are high performance, high availability servers, pre-optimized for latency with the set of chosen SNN's.	a slice fragments per slice	FEDP->SNN
12	Each SNN stores a complete <i>metadata2</i> map file needed to reassemble the slices, and the slice fragments assigned to that SNN. For speed, we do not need a very large number of SNN, with 5 SNN's optimizing the speed performance enhancements, but we want a much larger number of SNN's for security improvements, as slice fragments are spread throughout a larger number of SNN's. This could be at least 20 SNN's, or even 50 SNN's for extra security.	a slice fragments per slice	SNN
13	SNN's can be carefully rated as to their latency performance, relative to the invention's system to aid the user in defining which SNN's should be chosen for file dispersal.		SNN

FIG. 3 (CONT.)

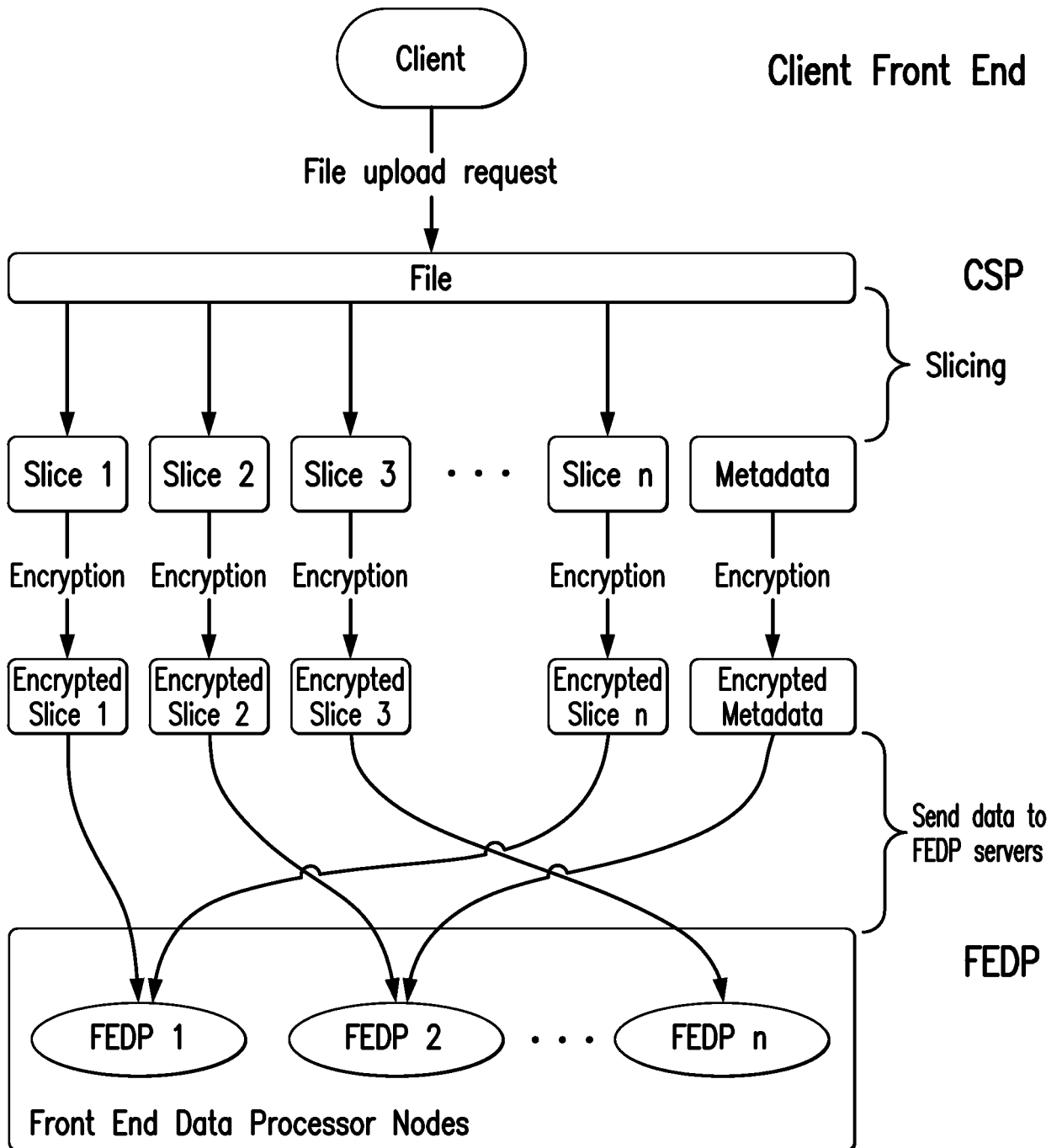


FIG. 4A

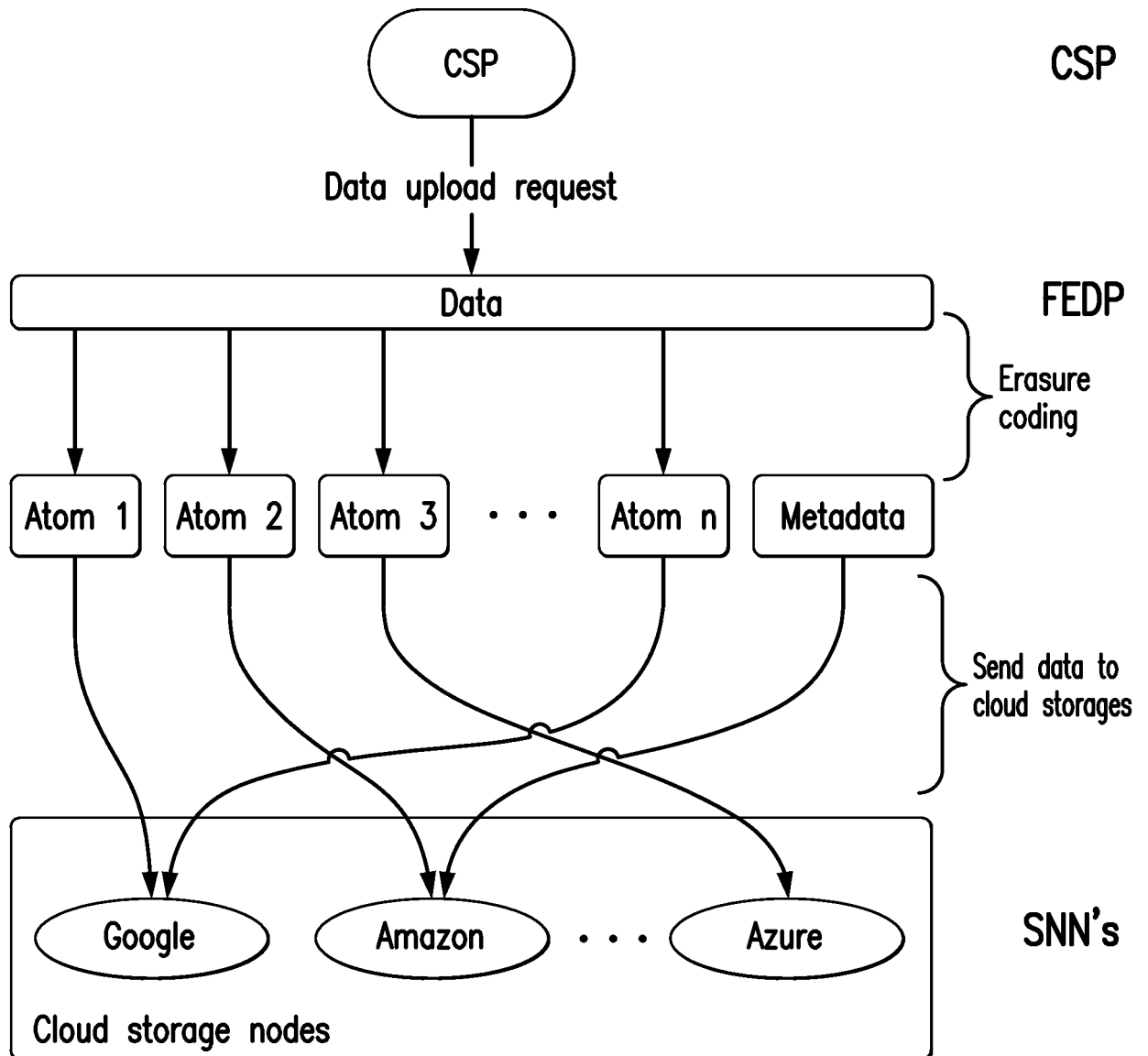


FIG. 4B

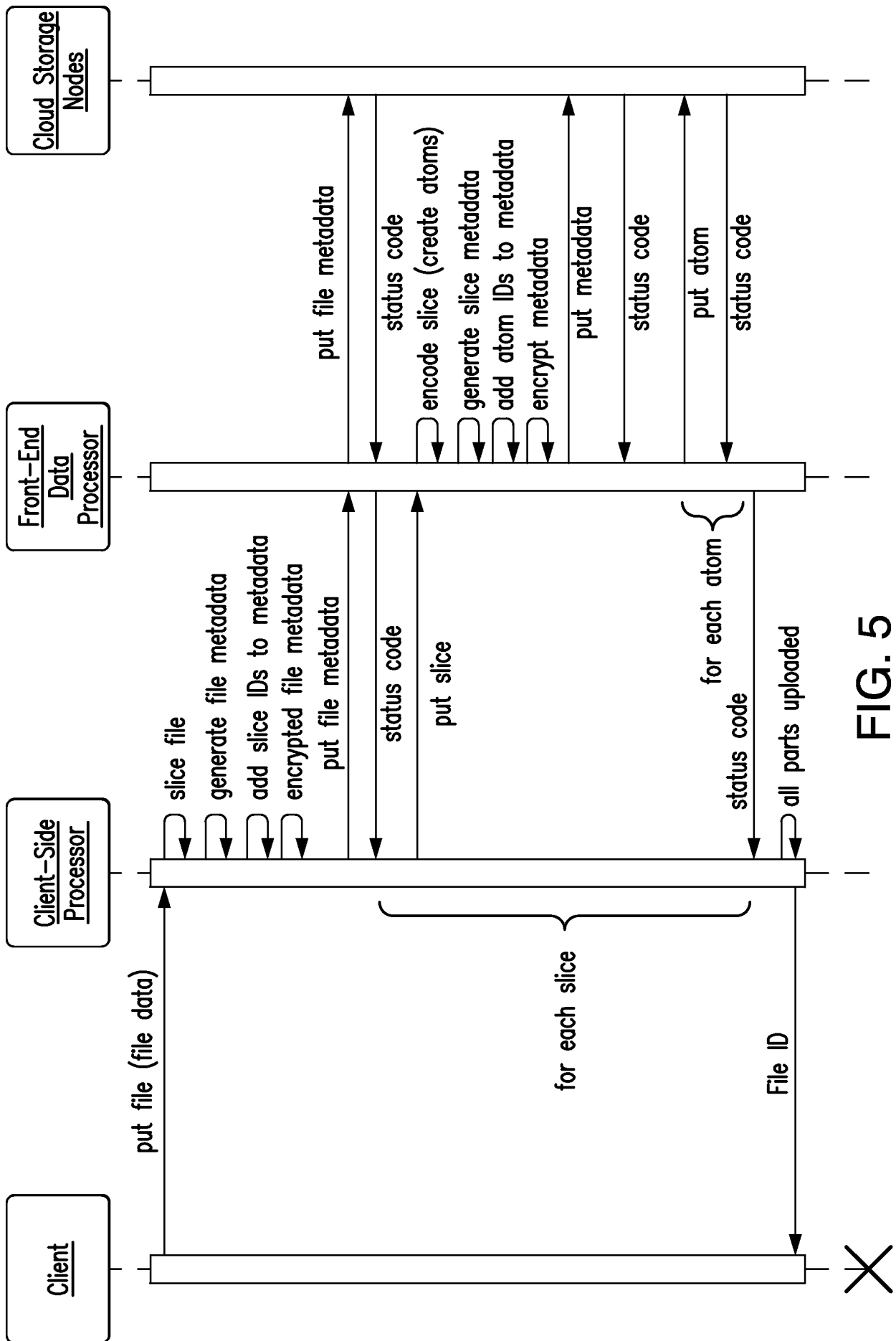


FIG. 5

1	Client app sends requests to download file to CSP for file that was previously uploaded. The client app sends the file id FID of the file to be downloaded to the CSP.	Uploaded file stored in SNN's	Client app
2	The CSP also receives <i>metadata1</i> which contains the map of slices to be reassembled into the whole file. This metadata file must be unencrypted using the client encryption key.	<i>metadata1</i> from client	CSP
3	The CSP needs to receive its file slices from the FEDP. The CSP sends the FEDP a request for file slice slice ID SID x .	Request slice SID x	CSP
4	THE FEDP receives a request for file slice SID x which causes it to send a request to one of the SNN's to receive metadata file <i>metadata2</i> . This metadata file must be unencrypted using the server's encryption key.	FEDP receives <i>metadata2</i>	FEDP
5	Once the FEDP has received and unencrypted <i>metadata2</i> , it has a map of where to find the slice fragments corresponding to this slice id SID x on the various SNN data centers.	FEDP requests slice fragments from slice SID	FEDP
6	The FEDP now sends out requests to all the SNN's which contain the slice fragments of file slice SID x , to retrieve these slice fragments according to their Fragment ID's. The FEDP makes use of the Erasure coding built into all the slice fragments to retrieve these fragments with error checking and error correction to insure that the slice fragments have been faithfully received by the FEDP.	SNN's transmit slice fragments to FEDP	FEDP
7	The FEDP decrypts all the slice fragments using the server's encryption key, and reassembles them into file slice SID x.	FEDP reassembles slice SID	FEDP
8	The FEDP transmits file slice SID x back to the CSP , as per the CSP's request, assuming all the error checks have passed.	FEDP transmits slice back to CSP	FEDP
9	The CSP takes the file slice SID x which it just received from the FEDP, and unencrypts it using the client's encryption key.	CSP unencrypts slice SID	CSP
10	The CSP uses the metadata file <i>metadata1</i> to request the next file slice SID y from the FEDP's.	CSP request next slice from FEDP	CSP
11	The FEDP repeats steps 4-8 above to assemble file slice y from the slice fragments stored on the SNN's until it has reassembled the requested file slice SID y, and sends this back to the CSP	CSP continues to request all the slices in <i>metadata1</i>	FEDP
12	The CSP unencrypts file slice SID y as above in step 9. The steps are repeated for each file slice SID contained in <i>metadata1</i> , with the CSP requesting assembled file slices from the FEDP, which assembles them from slice fragments dispersed among the SNN's.	CSP and FEDP repeat steps until all slices are retrieved by CSP.	CSP, FEDP

FIG. 6

1 3	When the CSP has received all the slices mapped in <i>metafile1</i> , it uses <i>metafile1</i> to reassemble the original, complete uploaded file, FID.	CSP reassembles all the slices into the original file FID, using <i>metadata1</i>	CSP
1 4	The CSP send the complete, reassembled, unencrypted file back to the client's data center, completing the client app's download request.	CSP transmits reassembled file to client data center	CSP

FIG. 6 (CONT.)

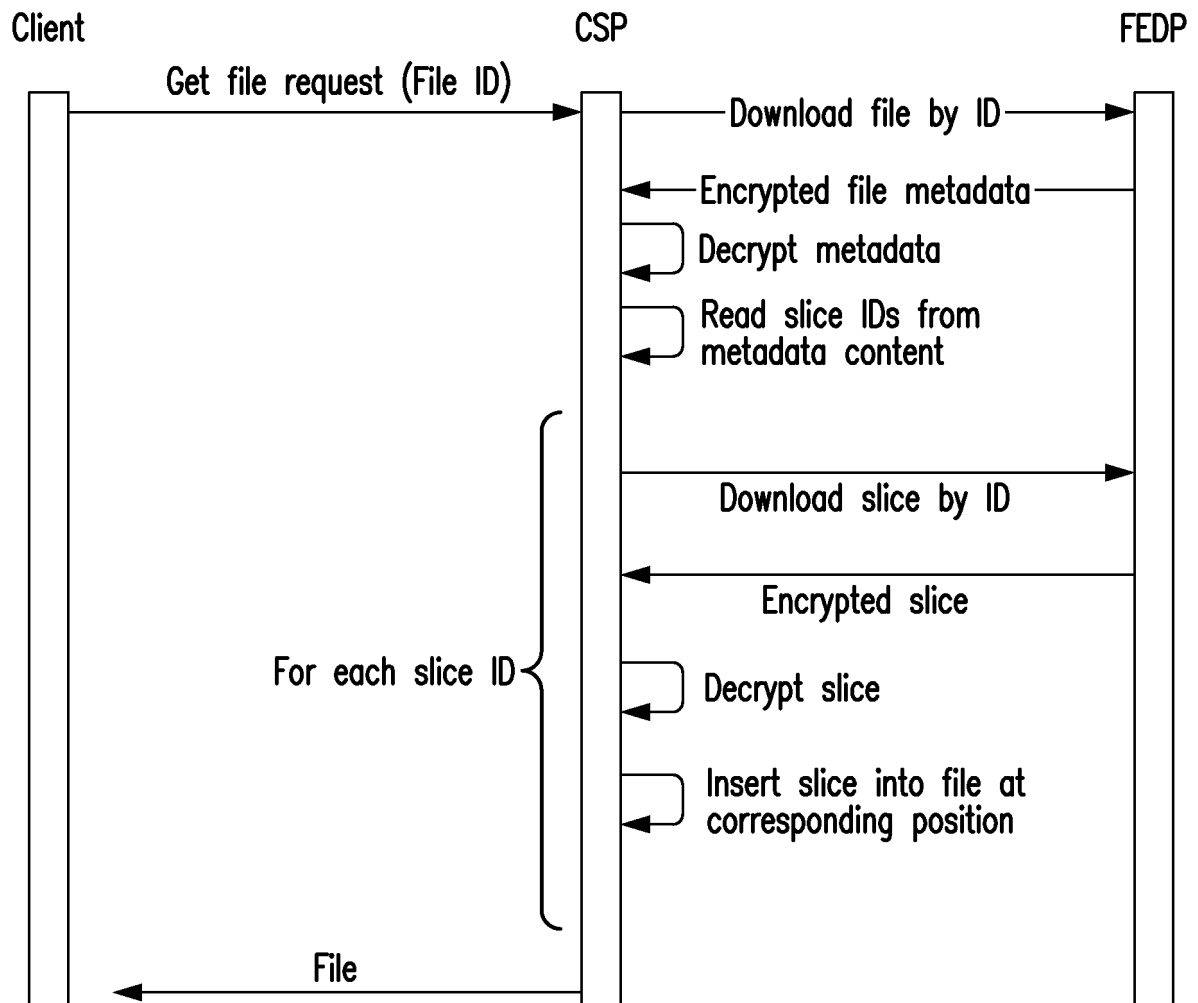


FIG. 7A

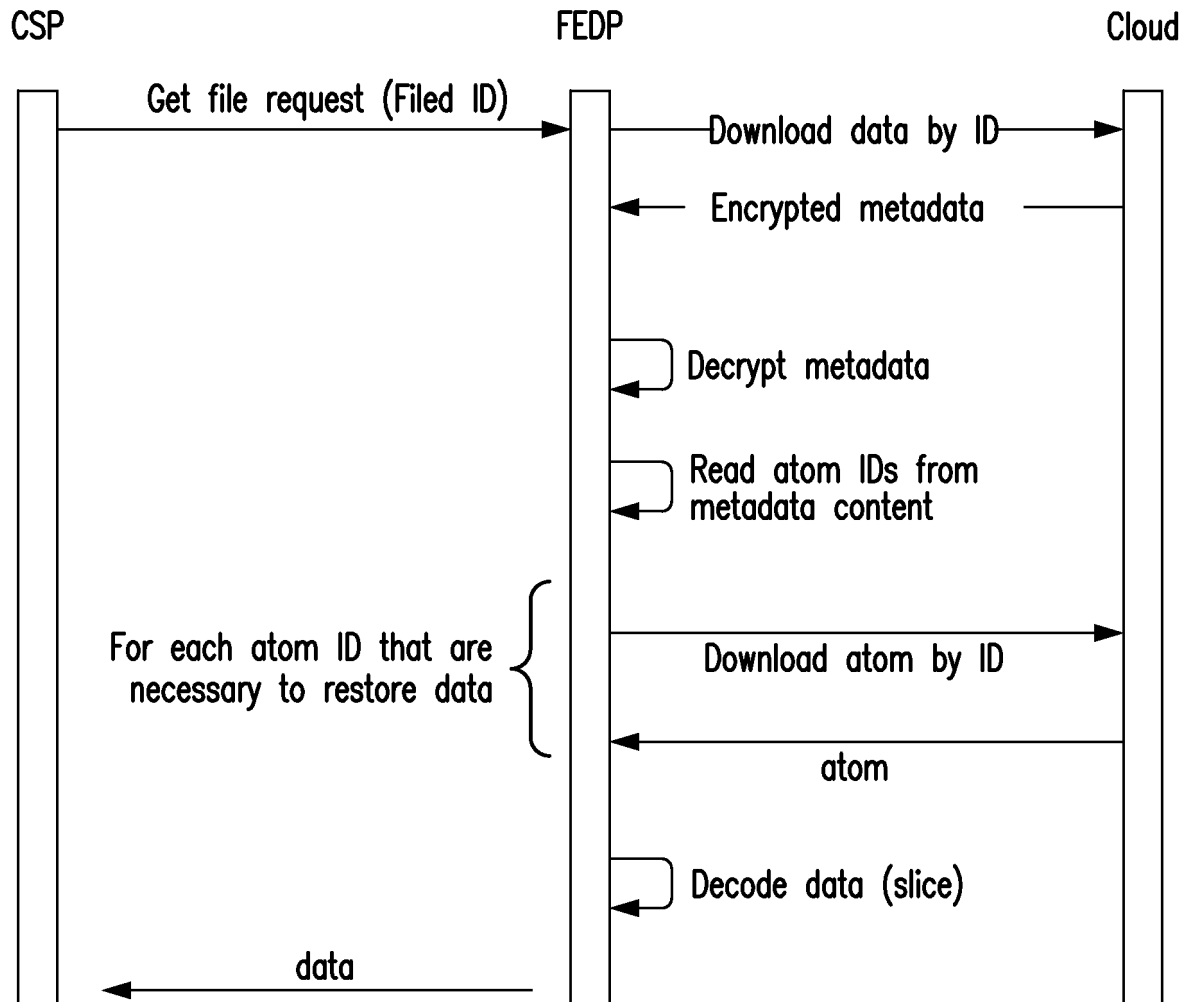


FIG. 7B

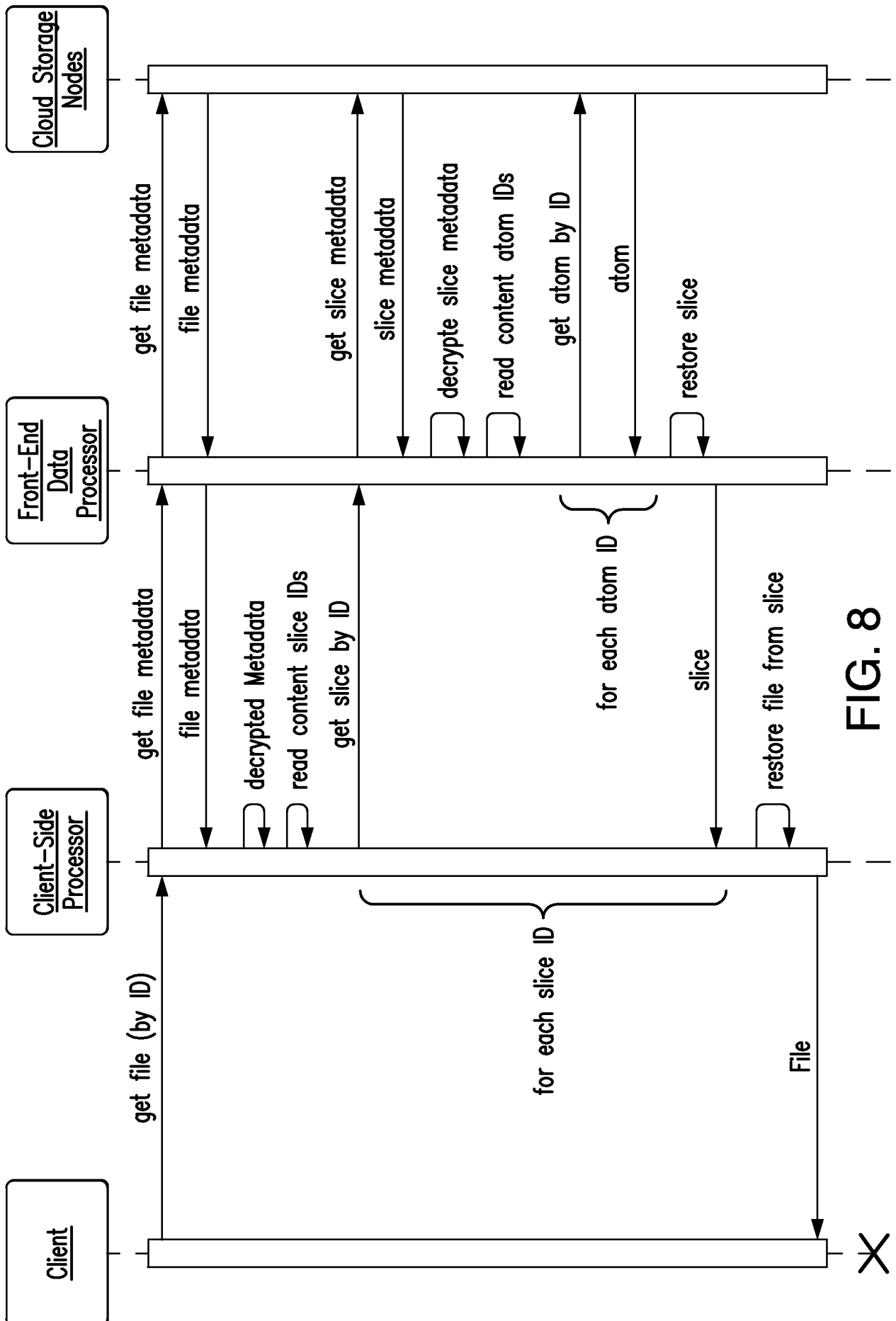


FIG. 8

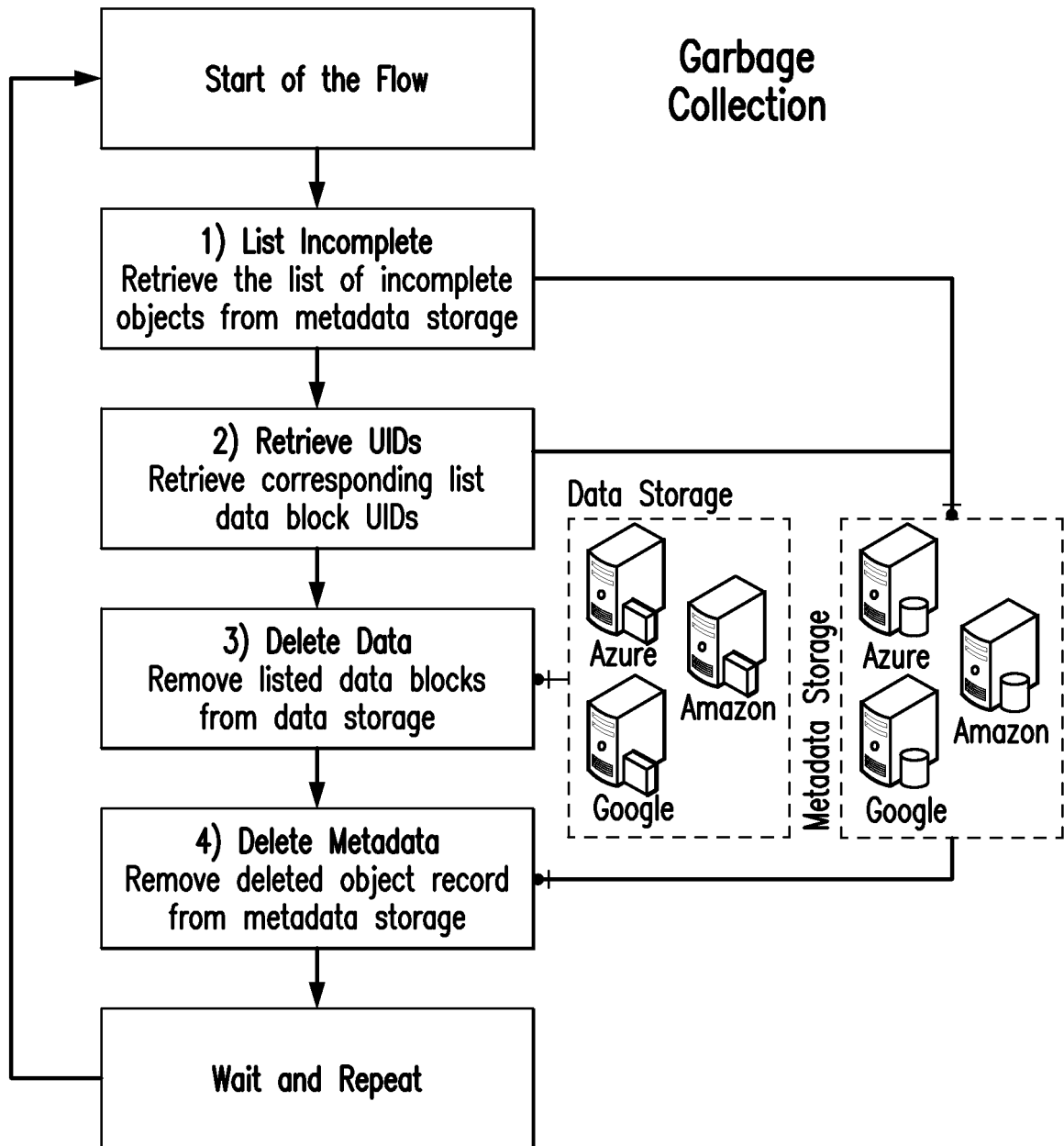


FIG. 9