

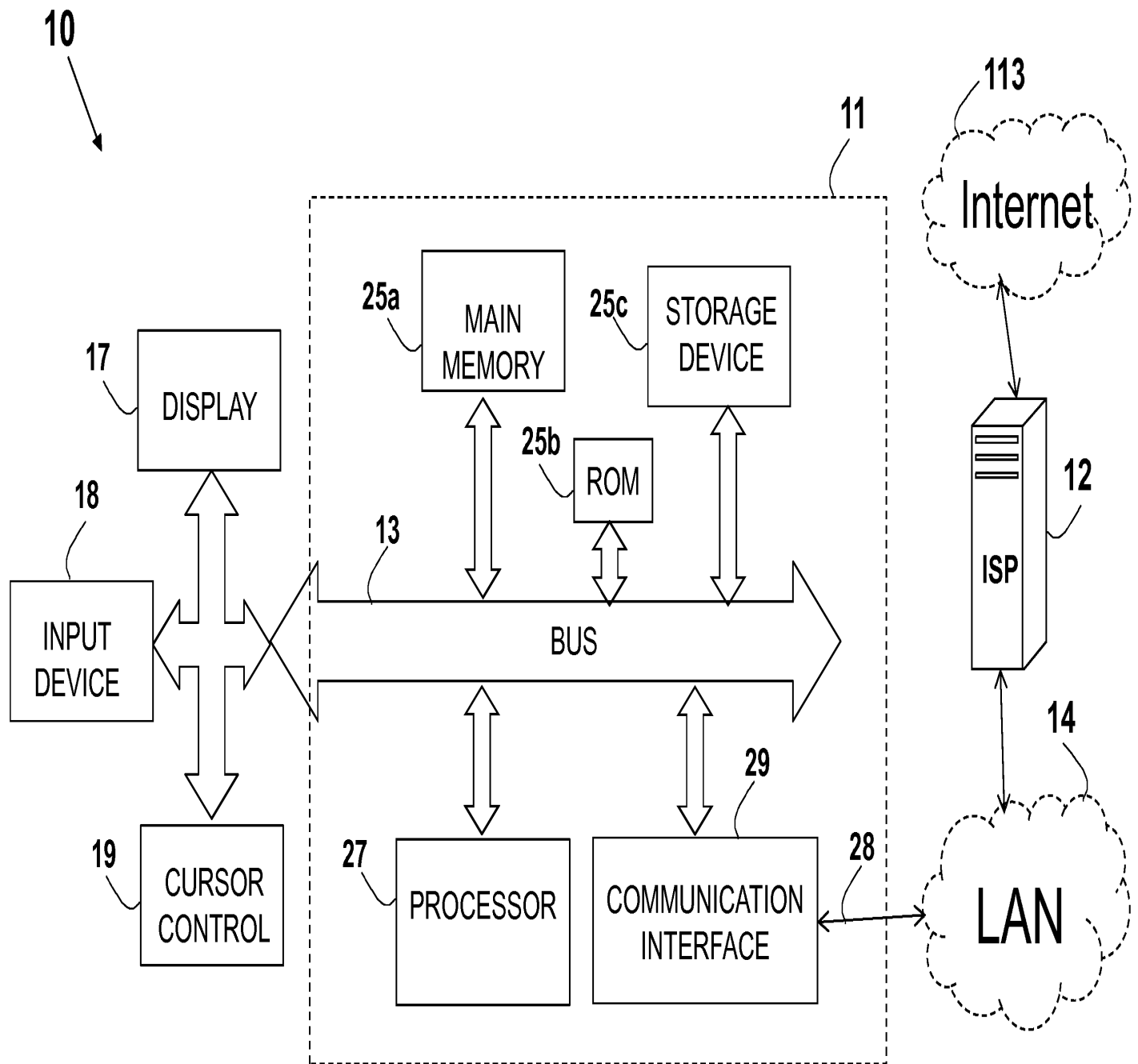


FIG. 1 (Prior Art)

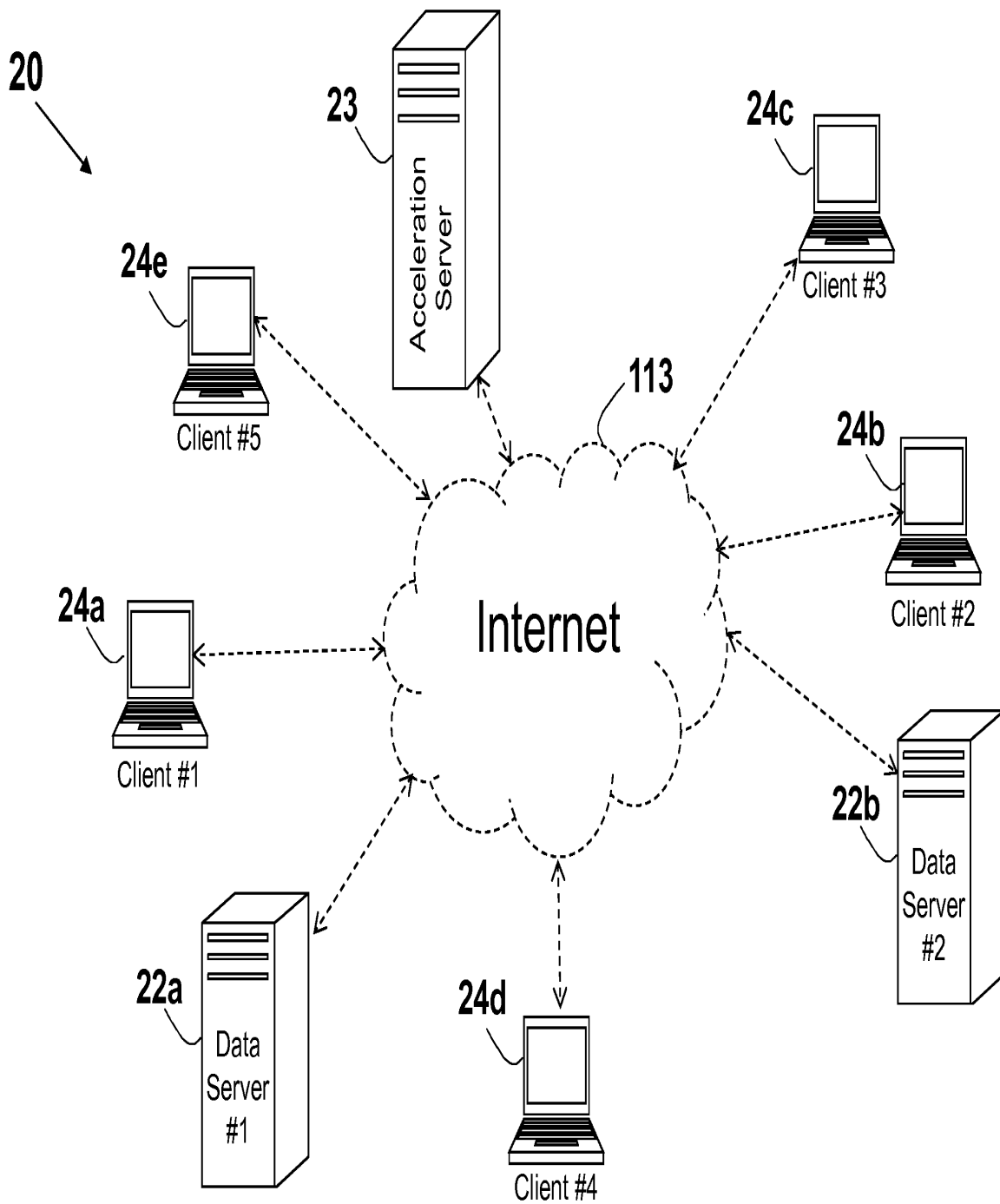


FIG. 2 (Prior Art)

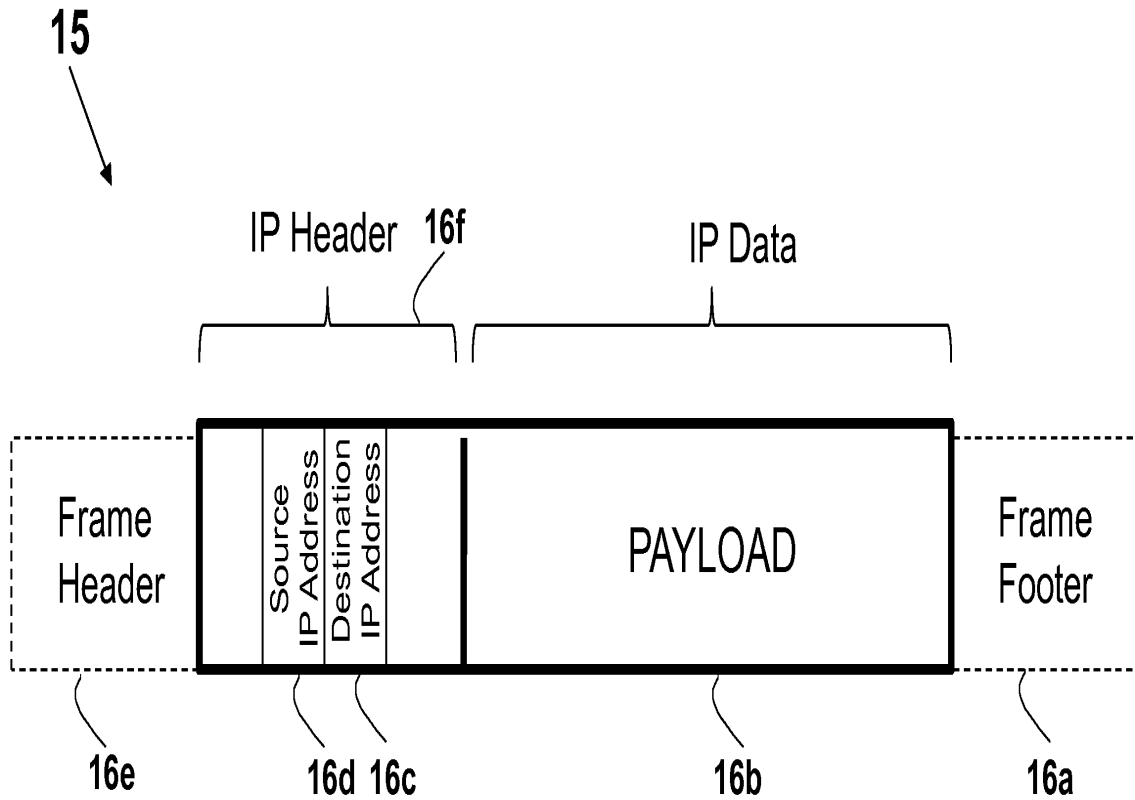


FIG. 2a (Prior Art)

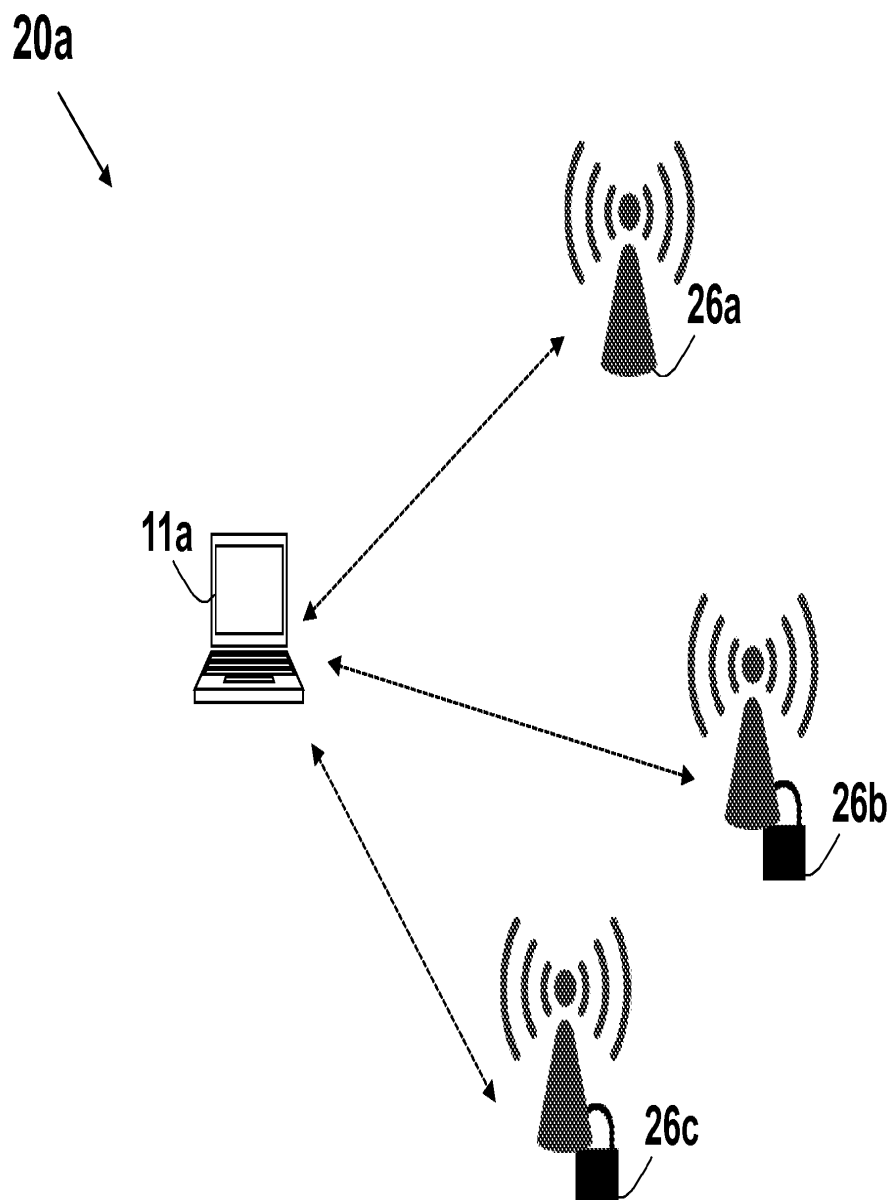


FIG. 2b (Prior Art)

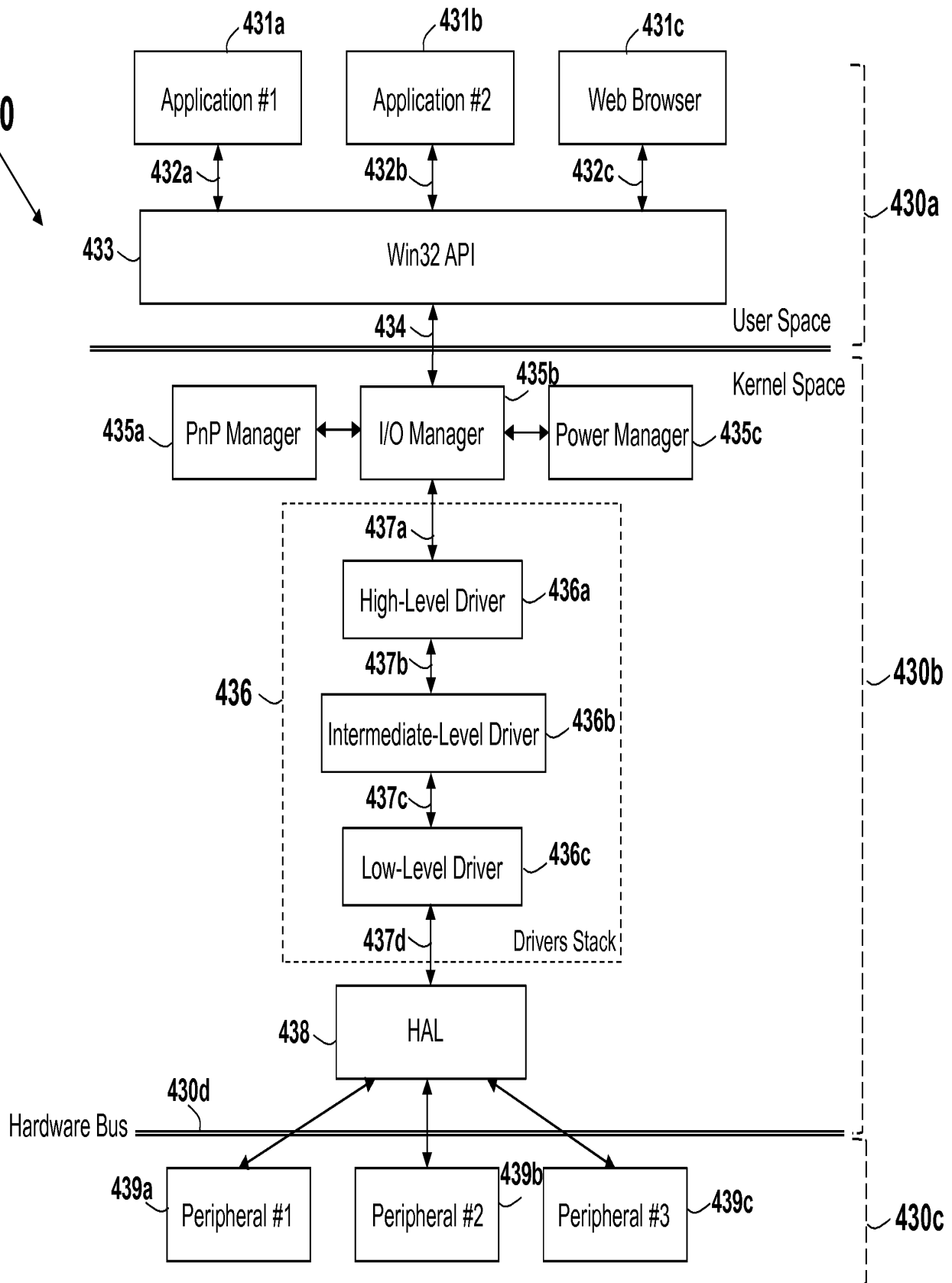


FIG. 3 (Prior Art)

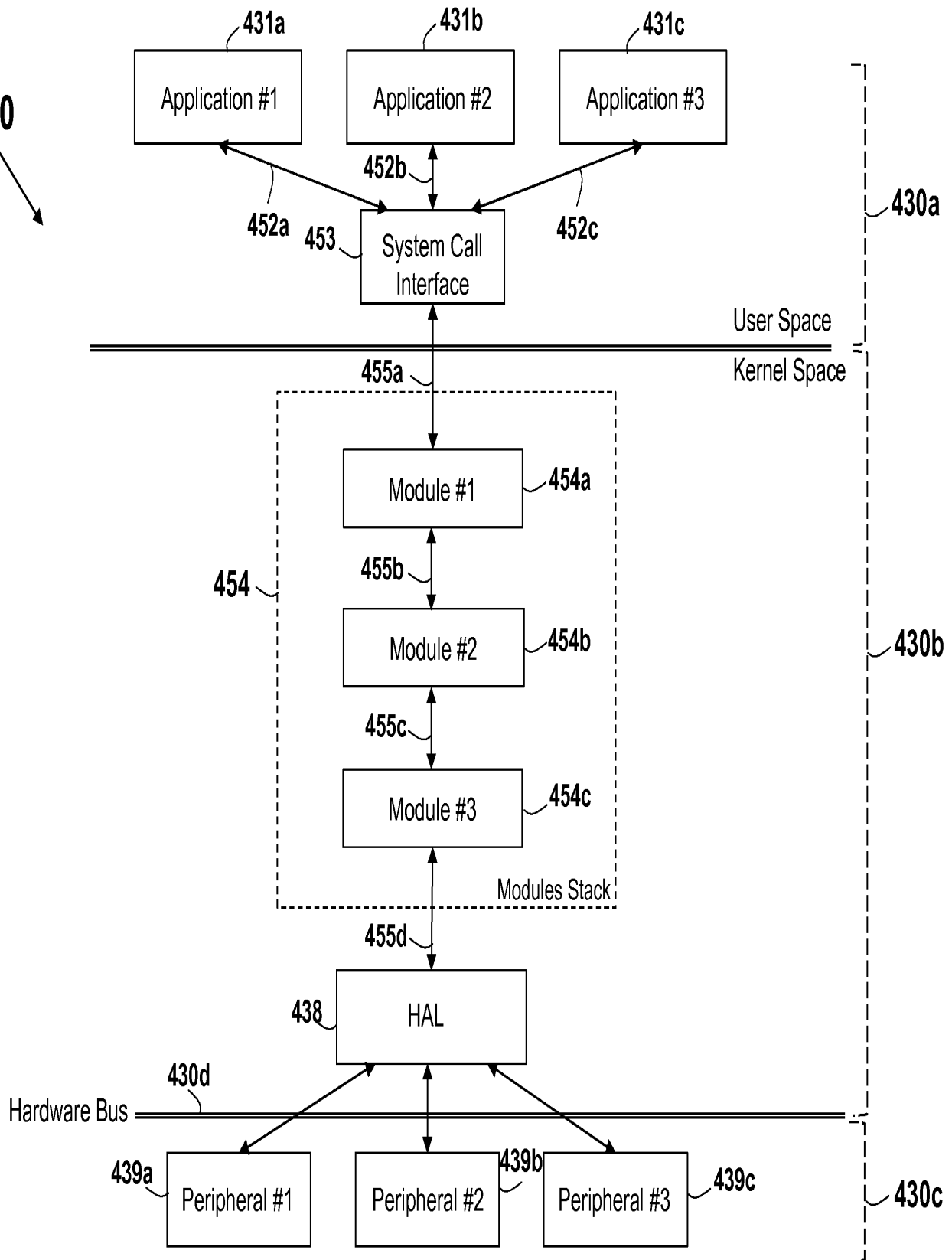


FIG. 3a (Prior Art)

470

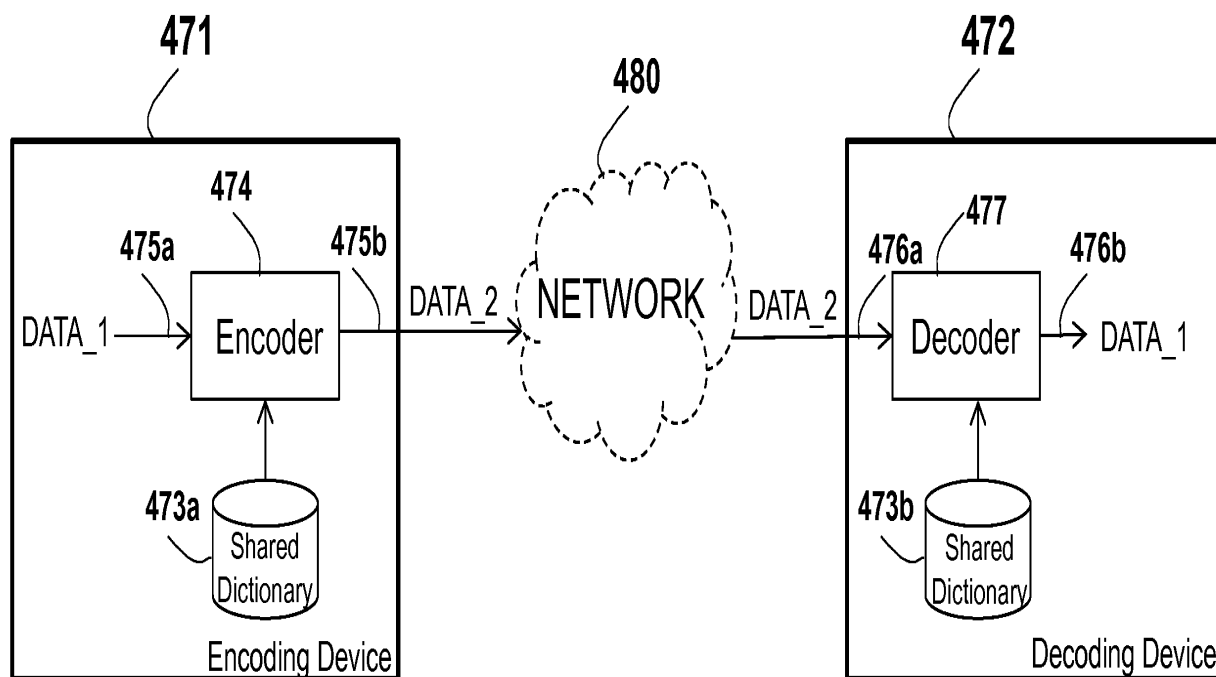


FIG. 4 (Prior Art)

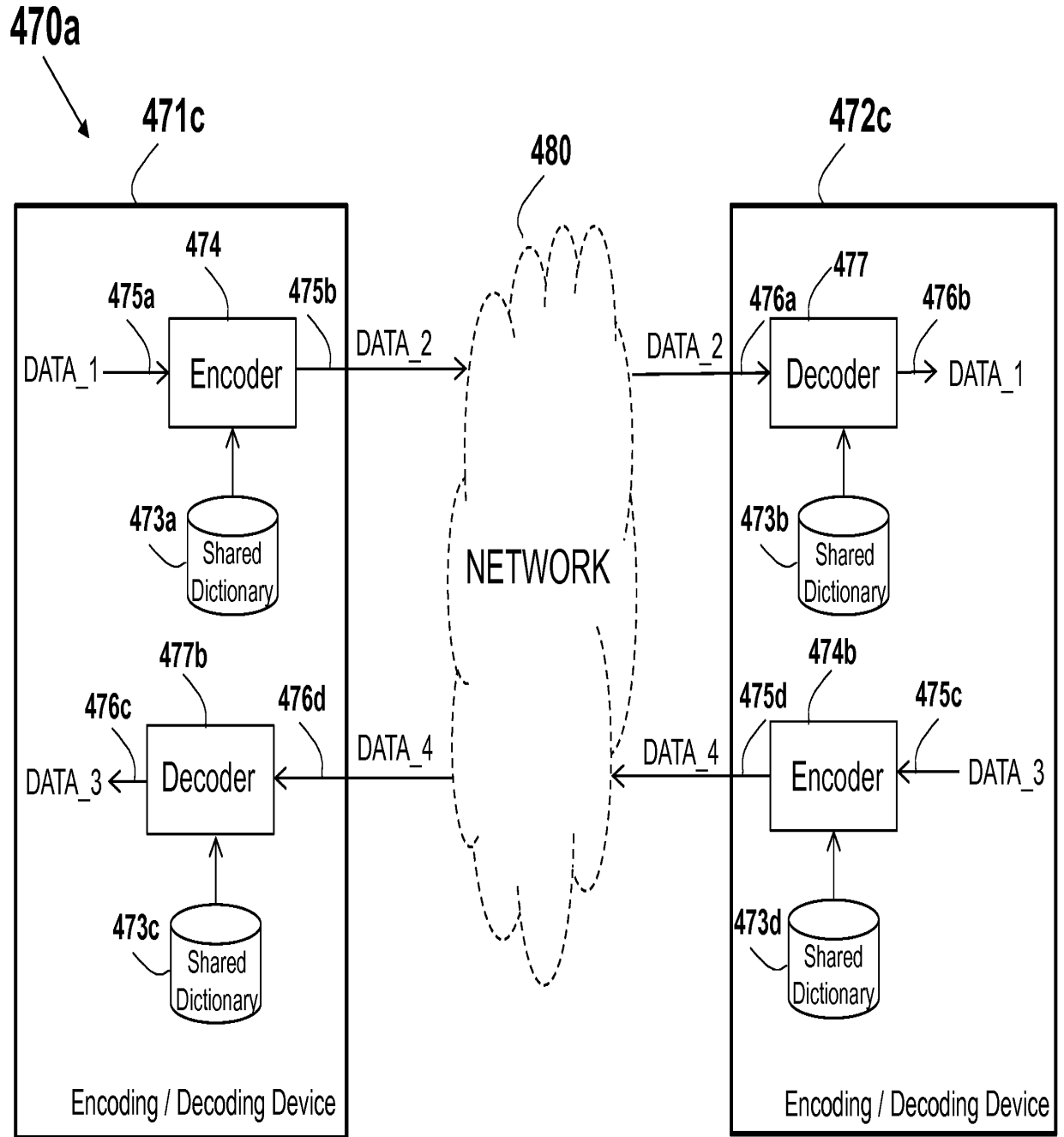


FIG. 4a (Prior Art)

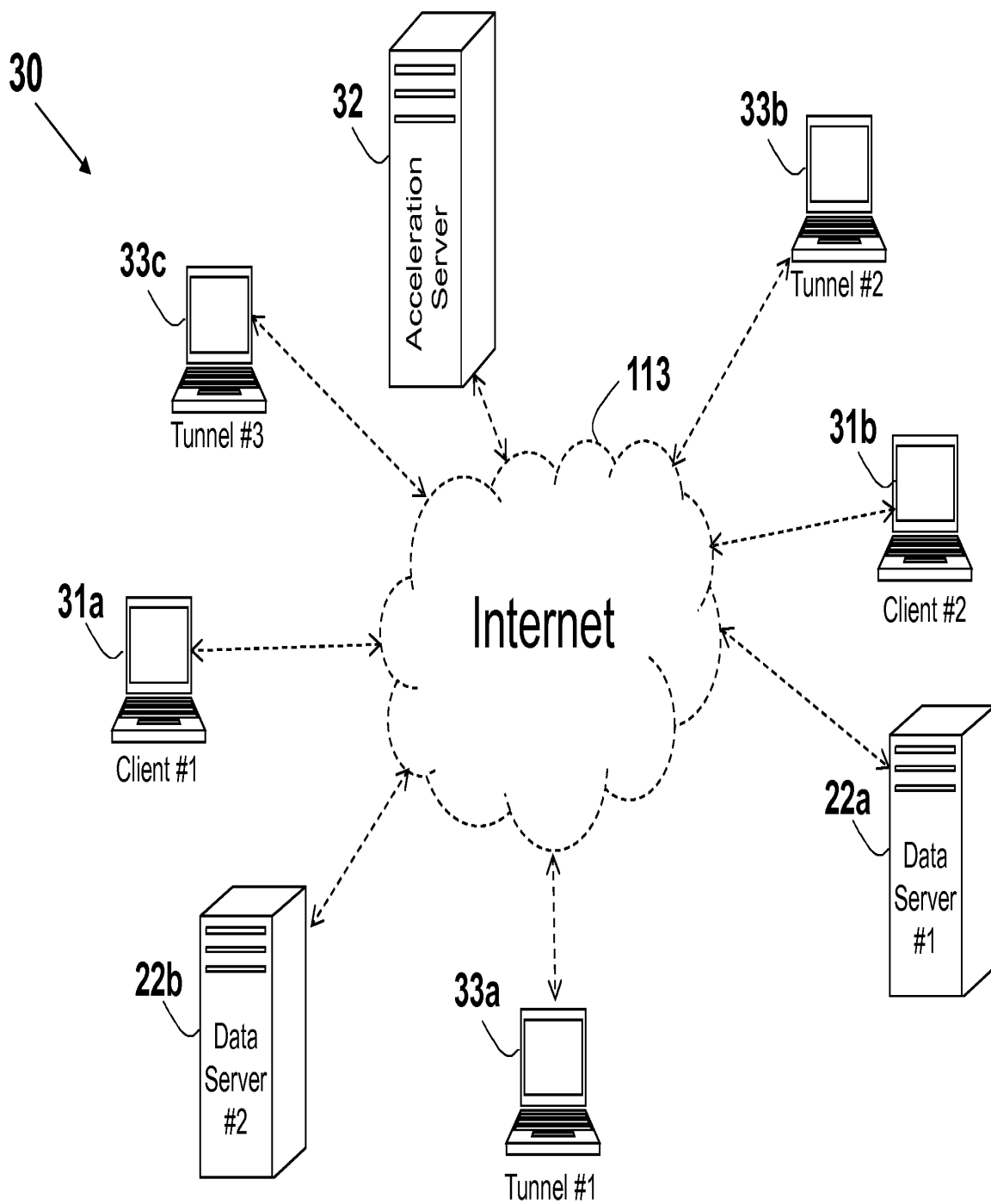


FIG. 5

40



41a



41b



41c



41d



41e



42 →

42a →

42b →

42c →

42d →

42e →

TYPE	IP ADDRESS	SIGN-IN DATE / TIME	LOCATION	STATUS
Tunnel	125.12.67.0	23/1 7:32	Boston, MA, USA	Online
Tunnel	109.23.78.5	23/1 8:55	Munich, Germany	Congested
Client	36.83.92.12	23/1 10:44	Sidney, Australia	Online
Client	125.66.69.73	24/1 15:34	Tel-Aviv, Israel	Offline
Client / Tunnel	103.52.25.73	24/1 20:42	Cairo, Egypt	Congested

FIG. 5a

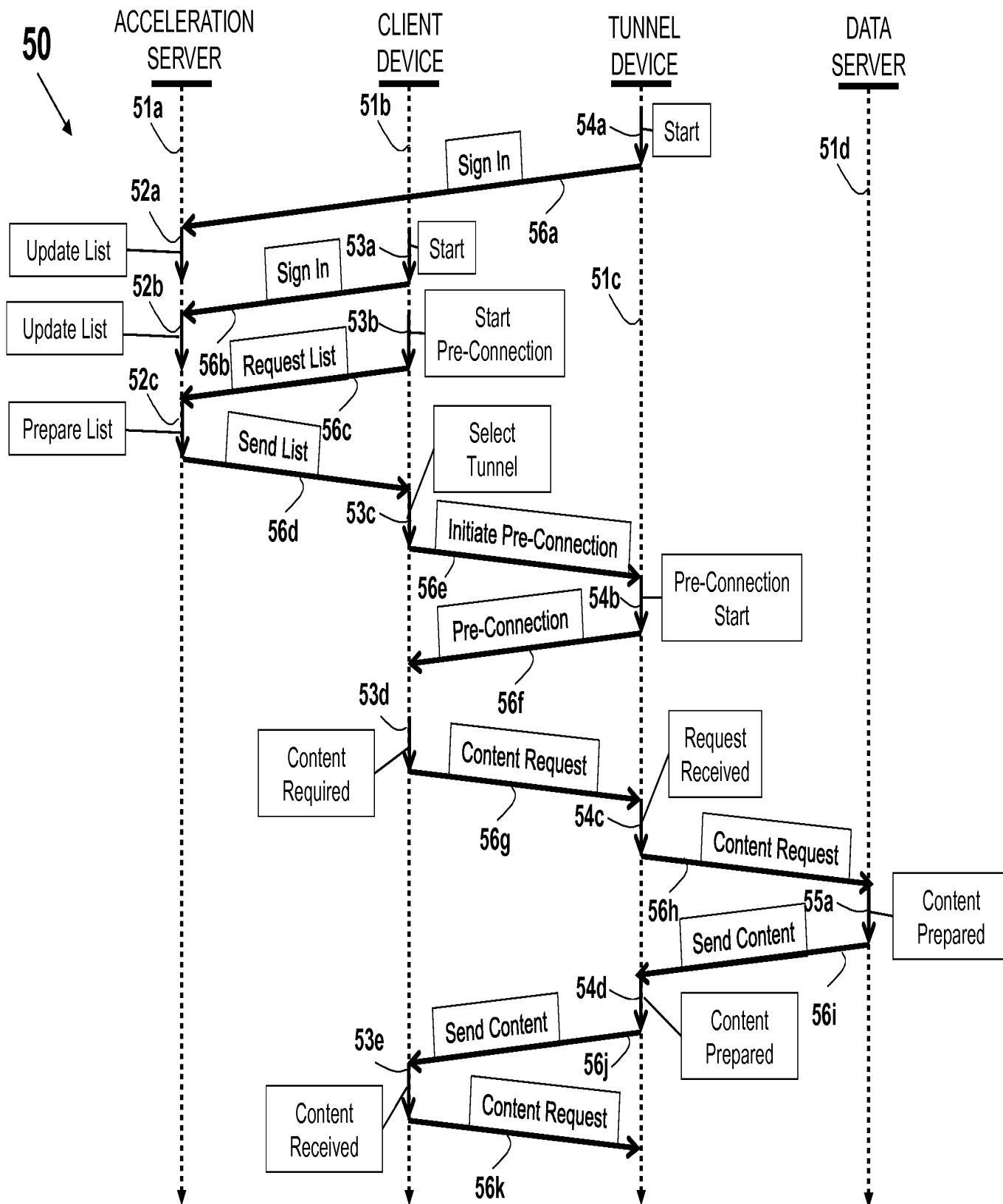


FIG. 5b

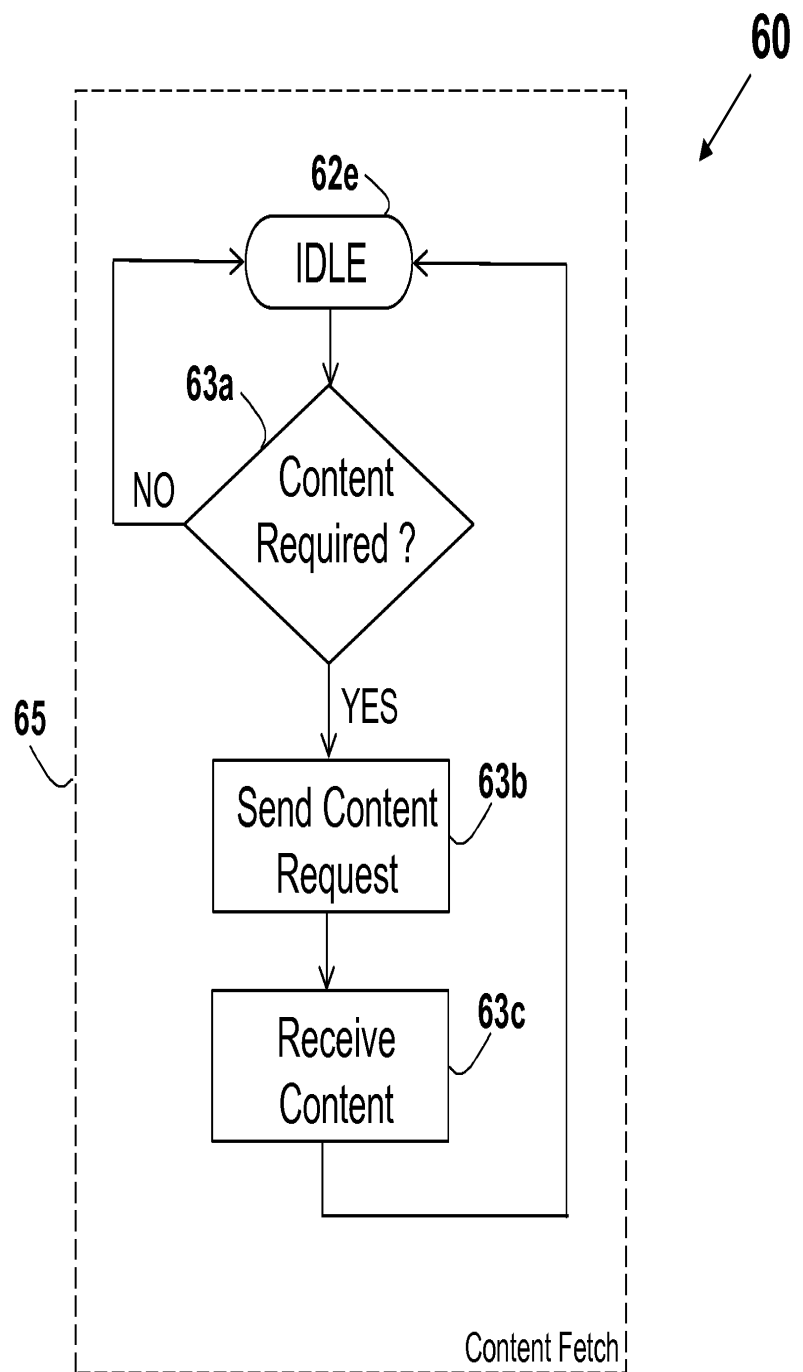
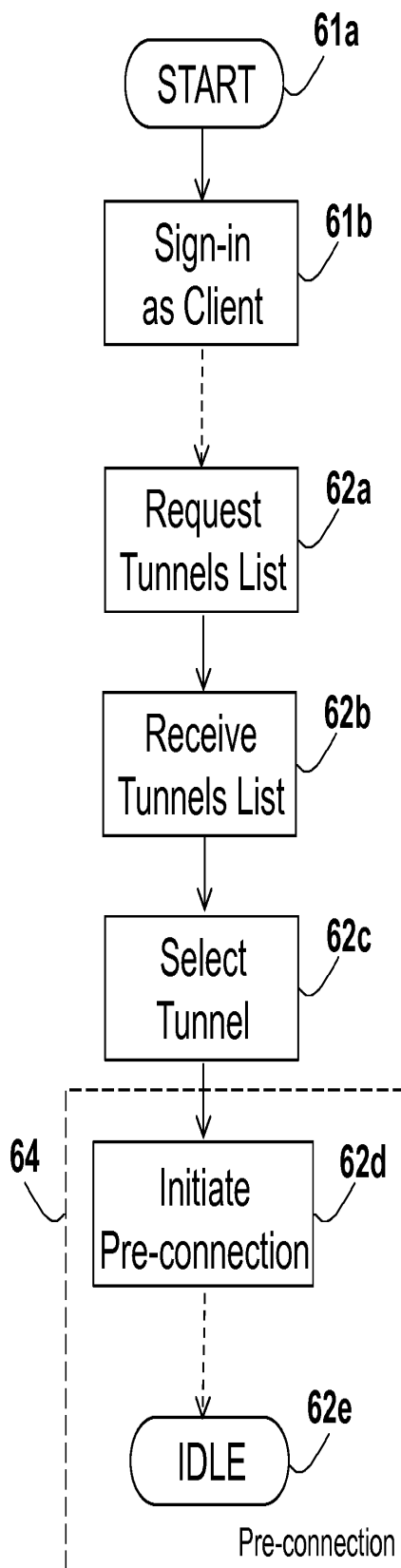


FIG. 6

70

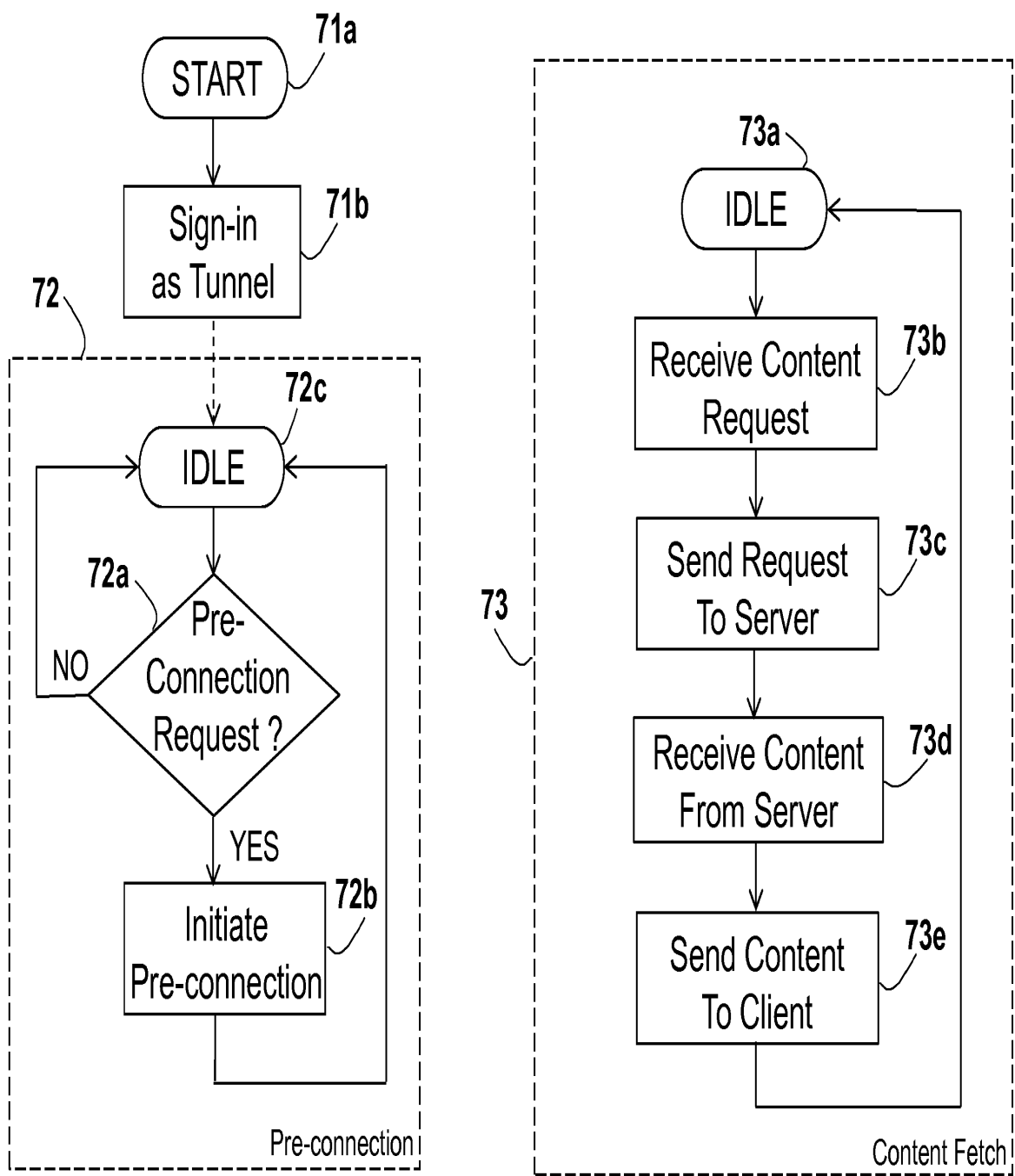


FIG. 7

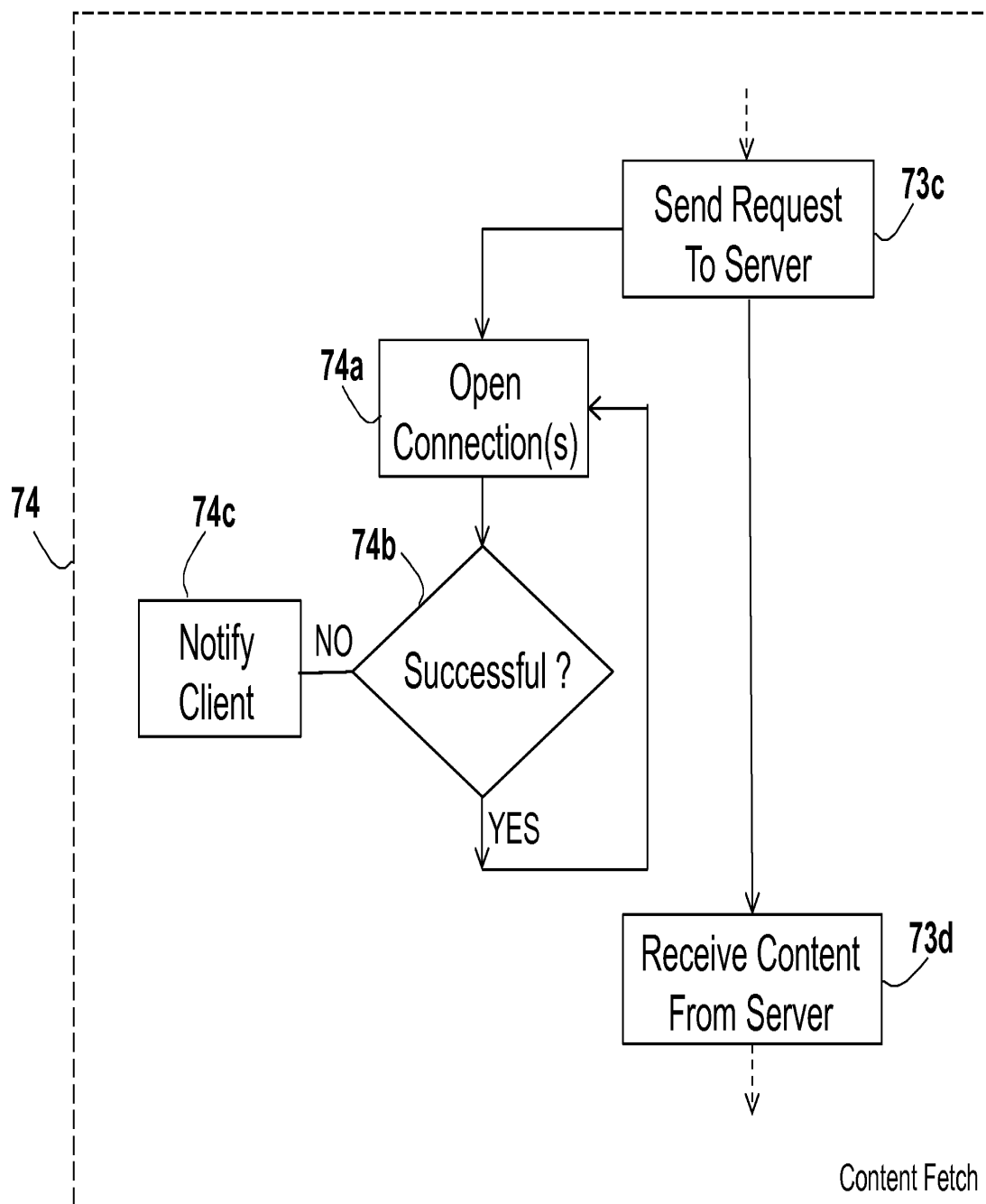


FIG. 7a

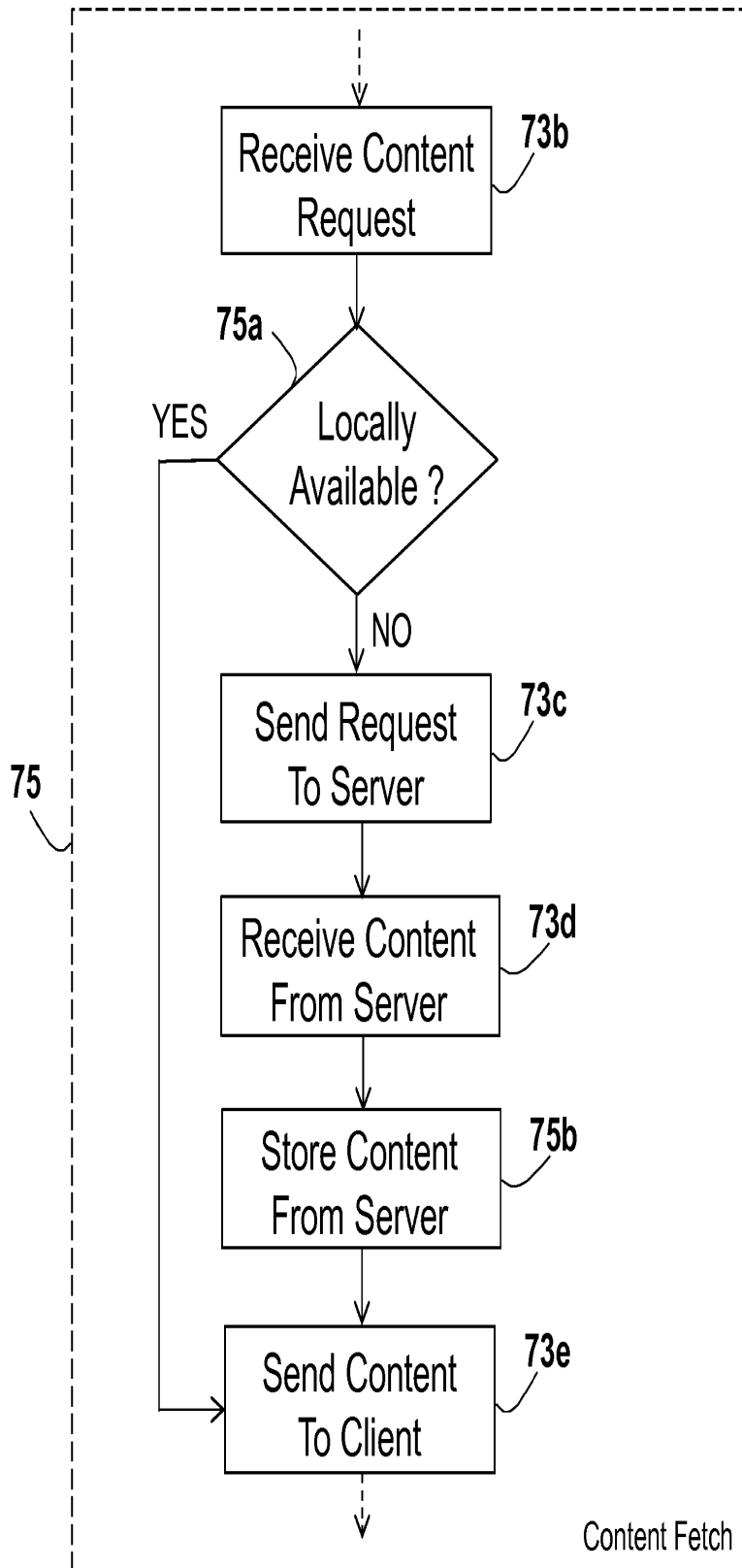


FIG. 7b

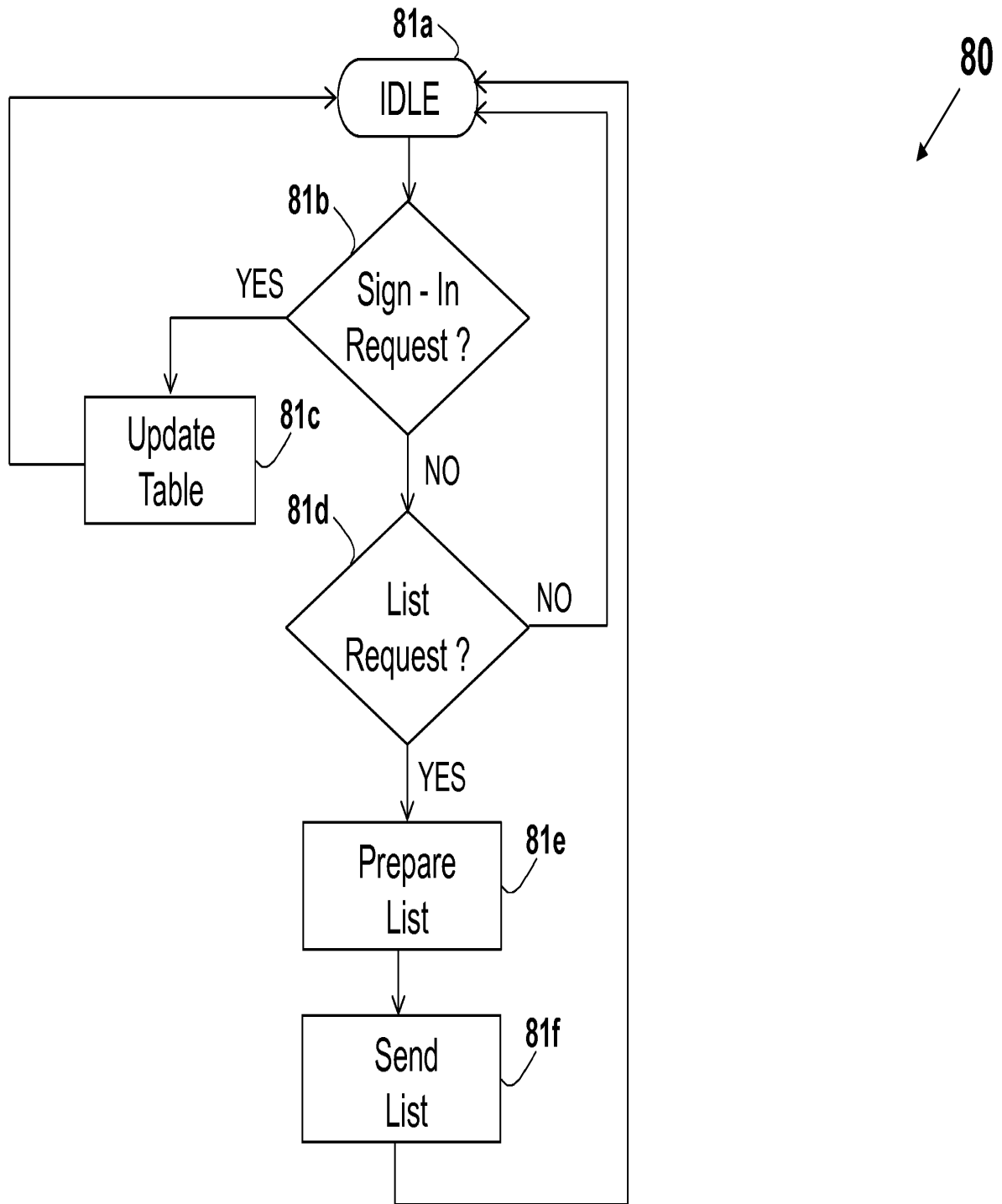
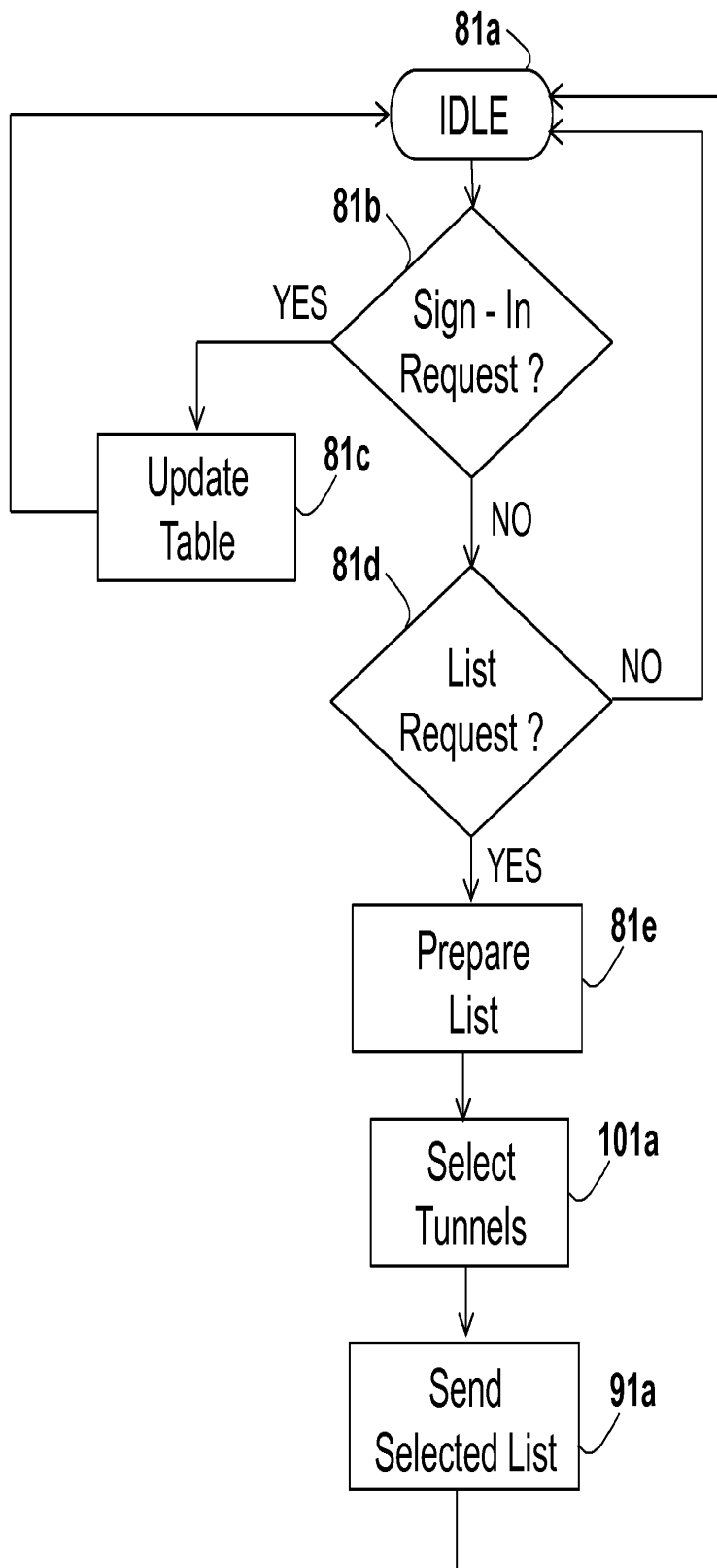


FIG. 8



90

FIG. 9

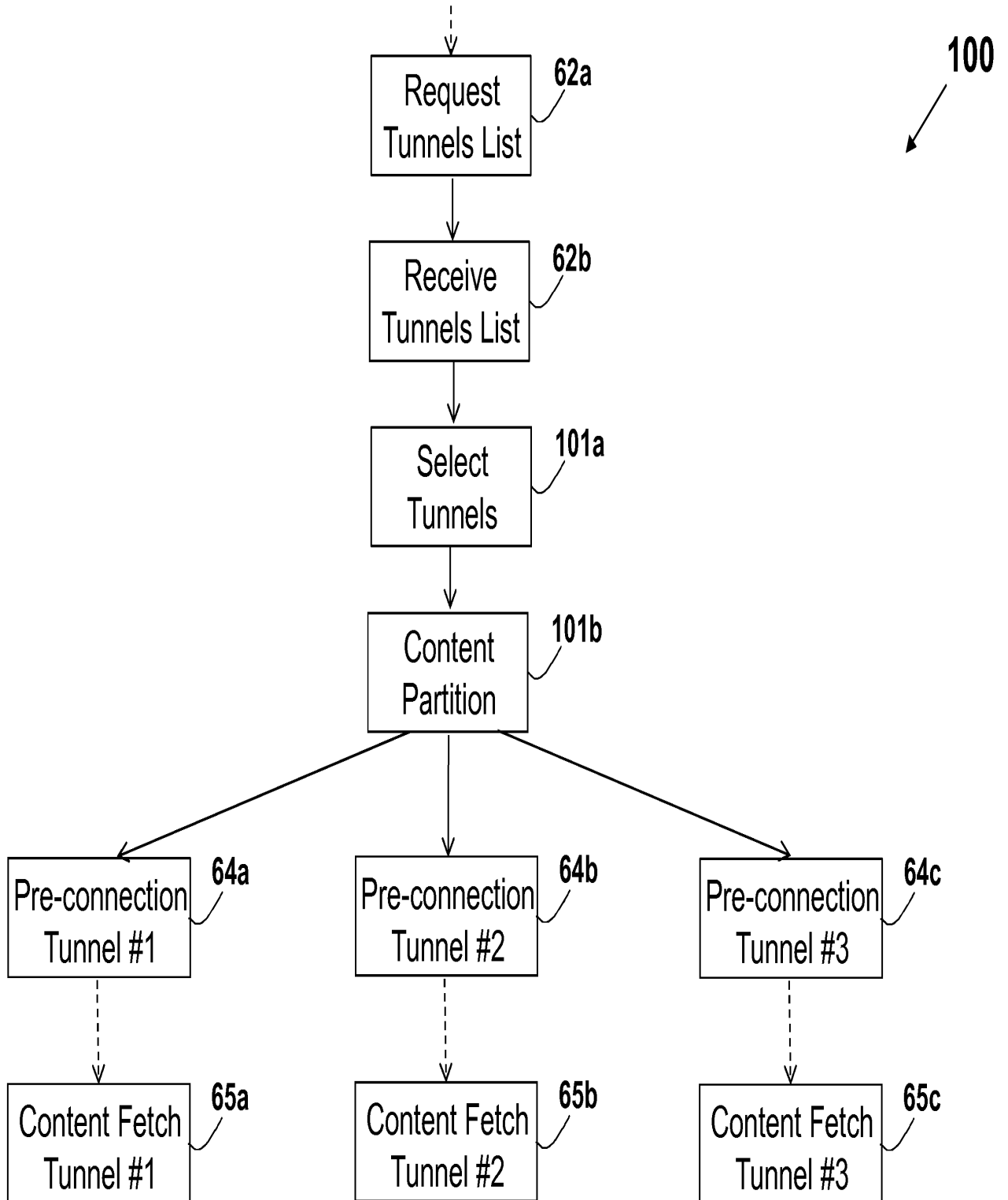


FIG. 10

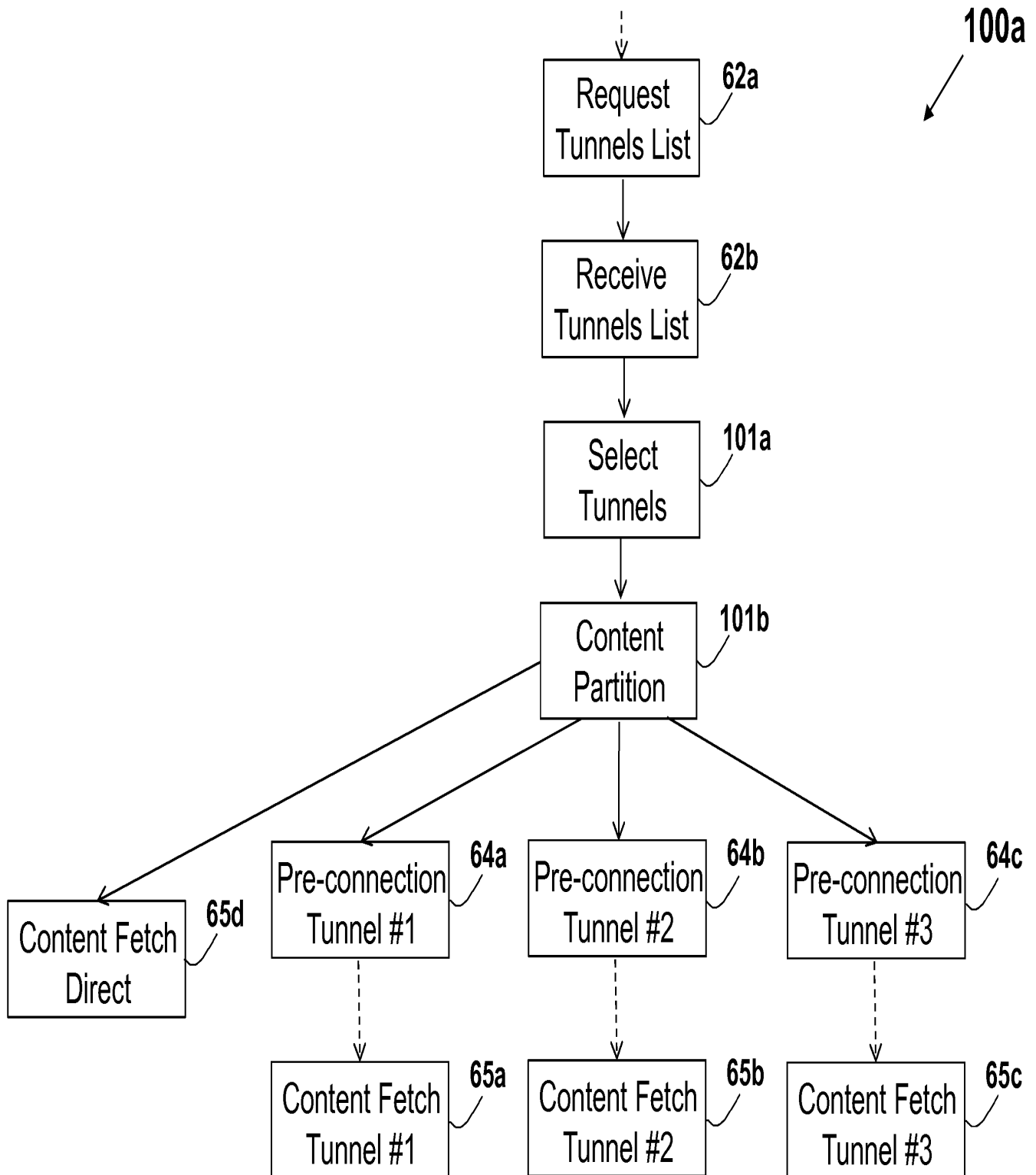


FIG. 10a

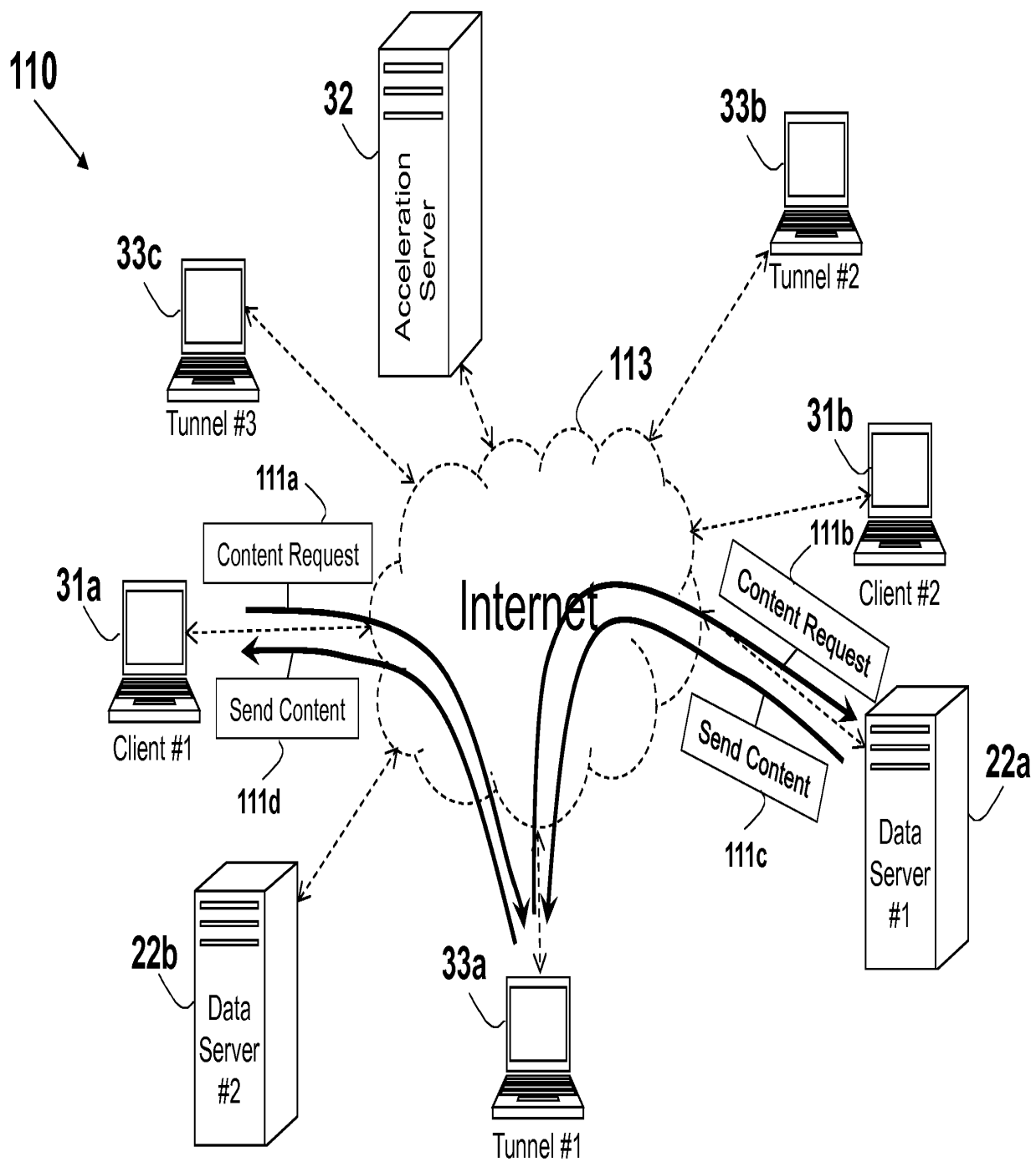


FIG. 11

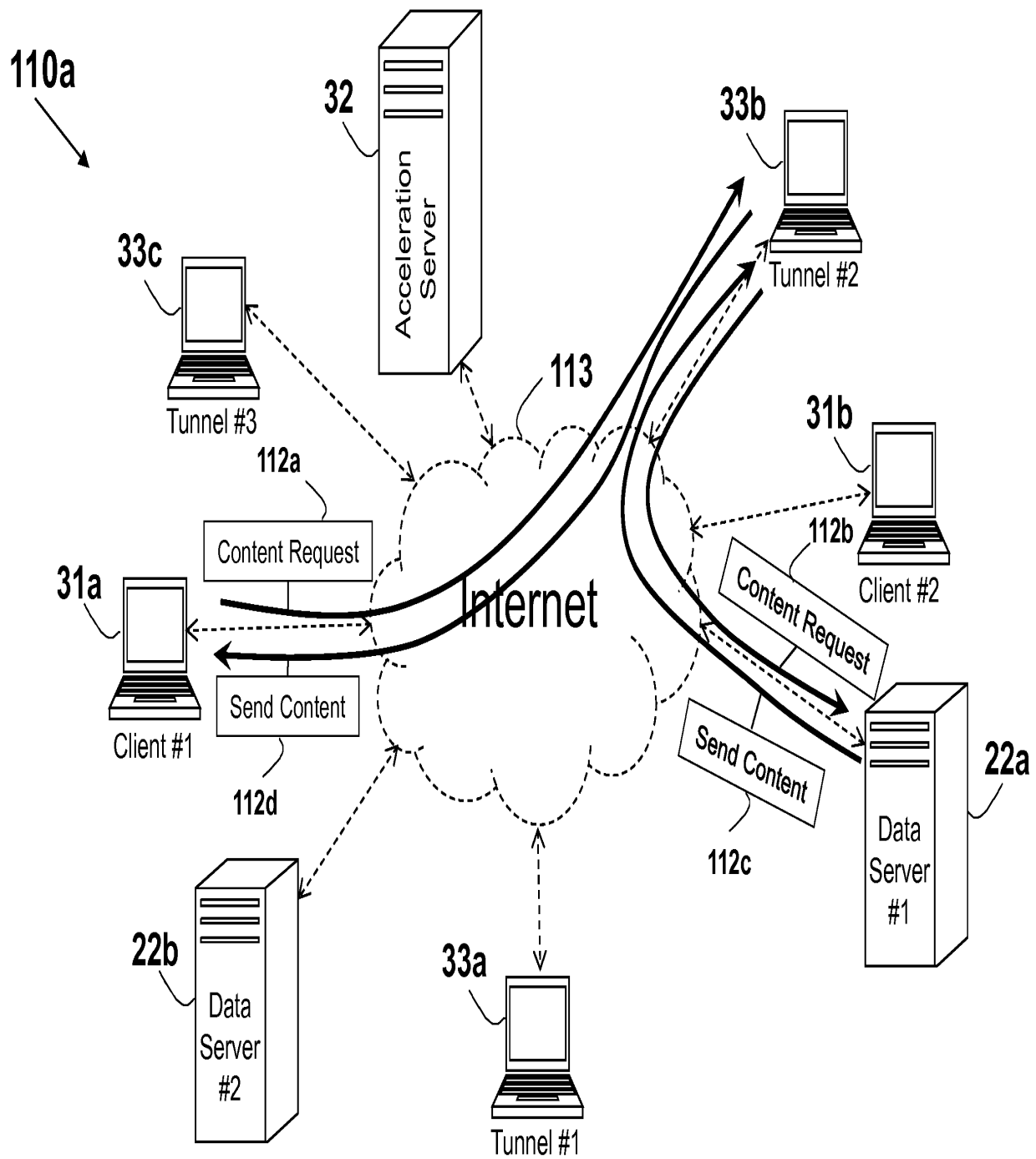


FIG. 11a

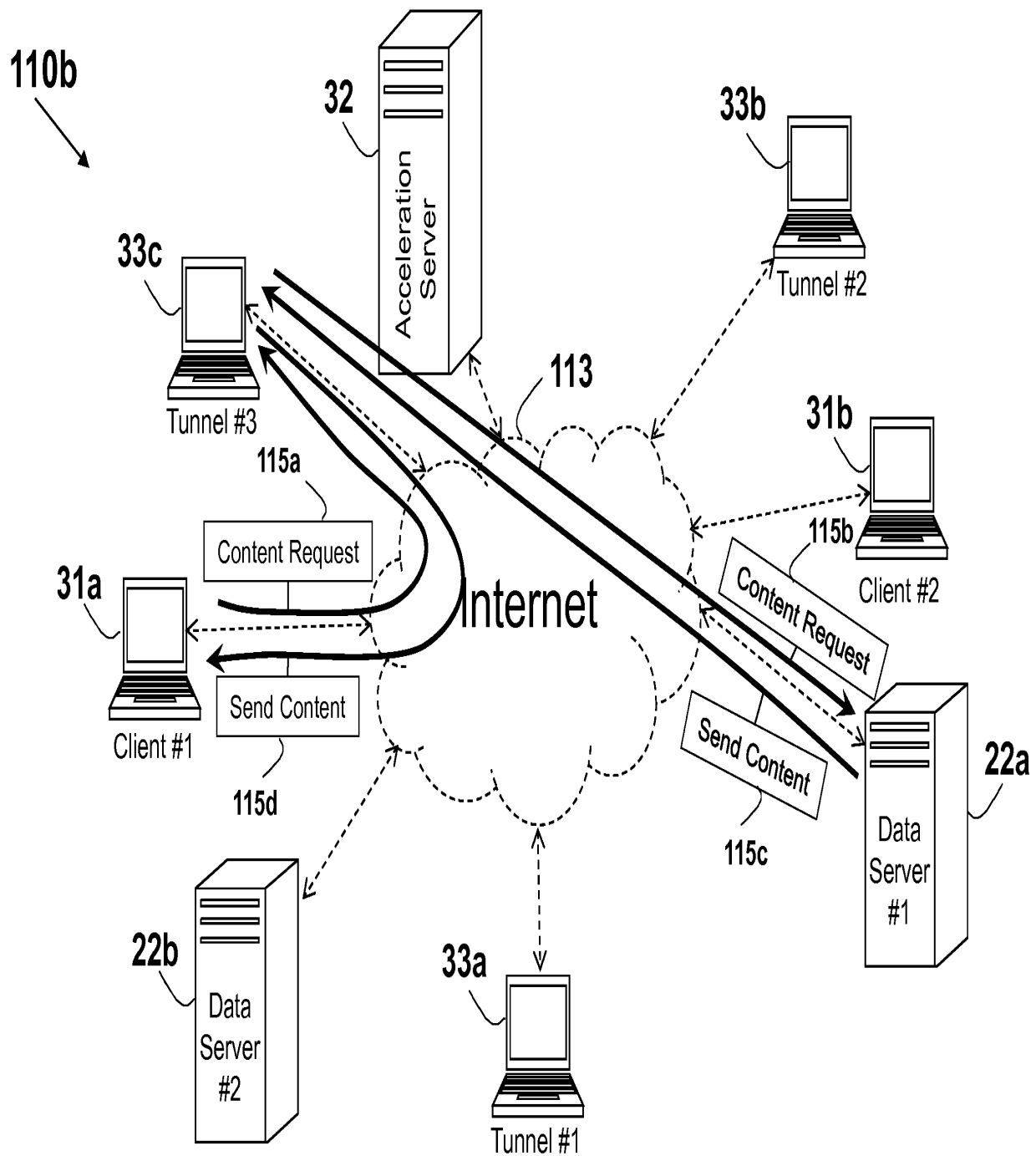


FIG. 11b

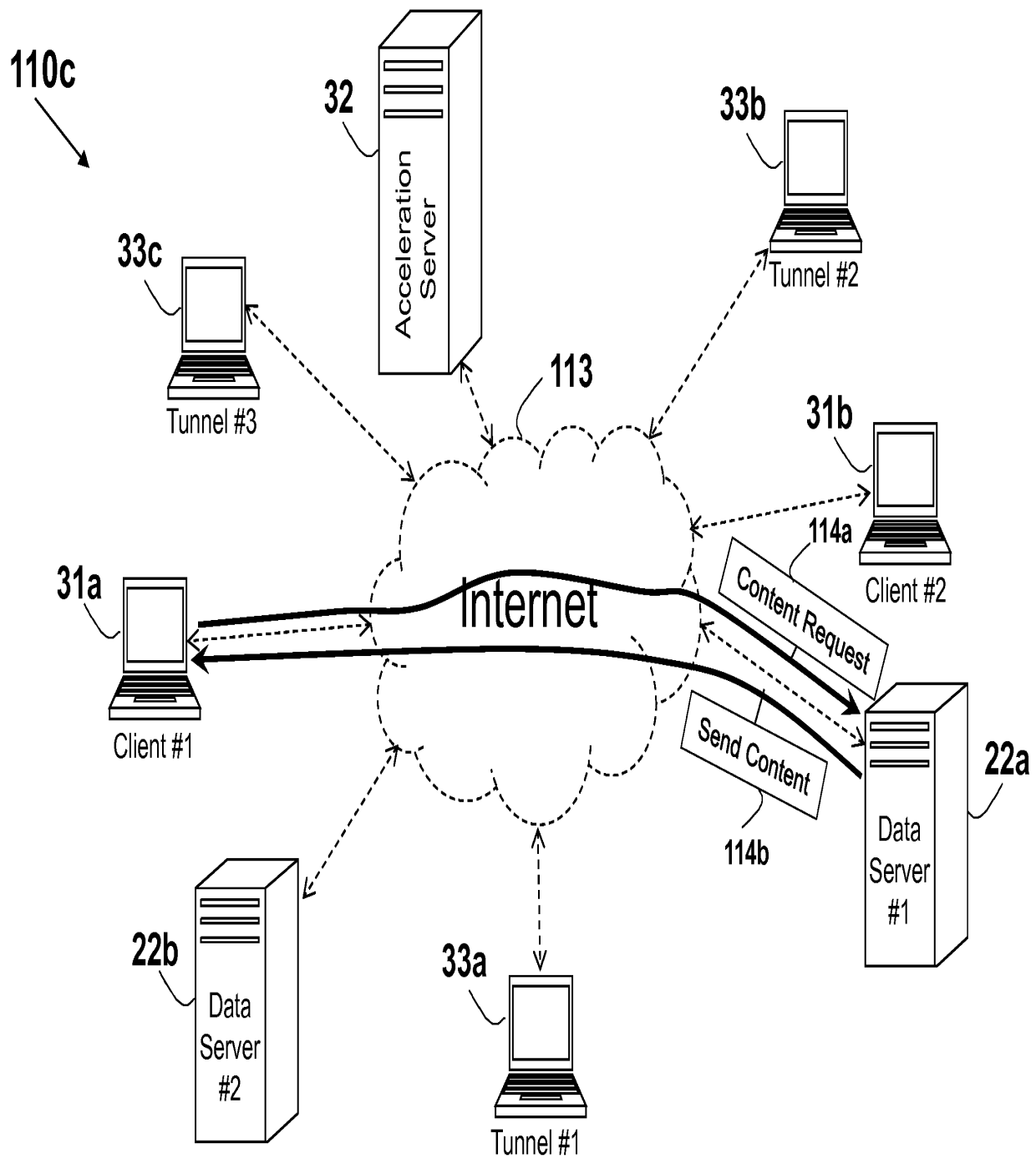


FIG. 11c

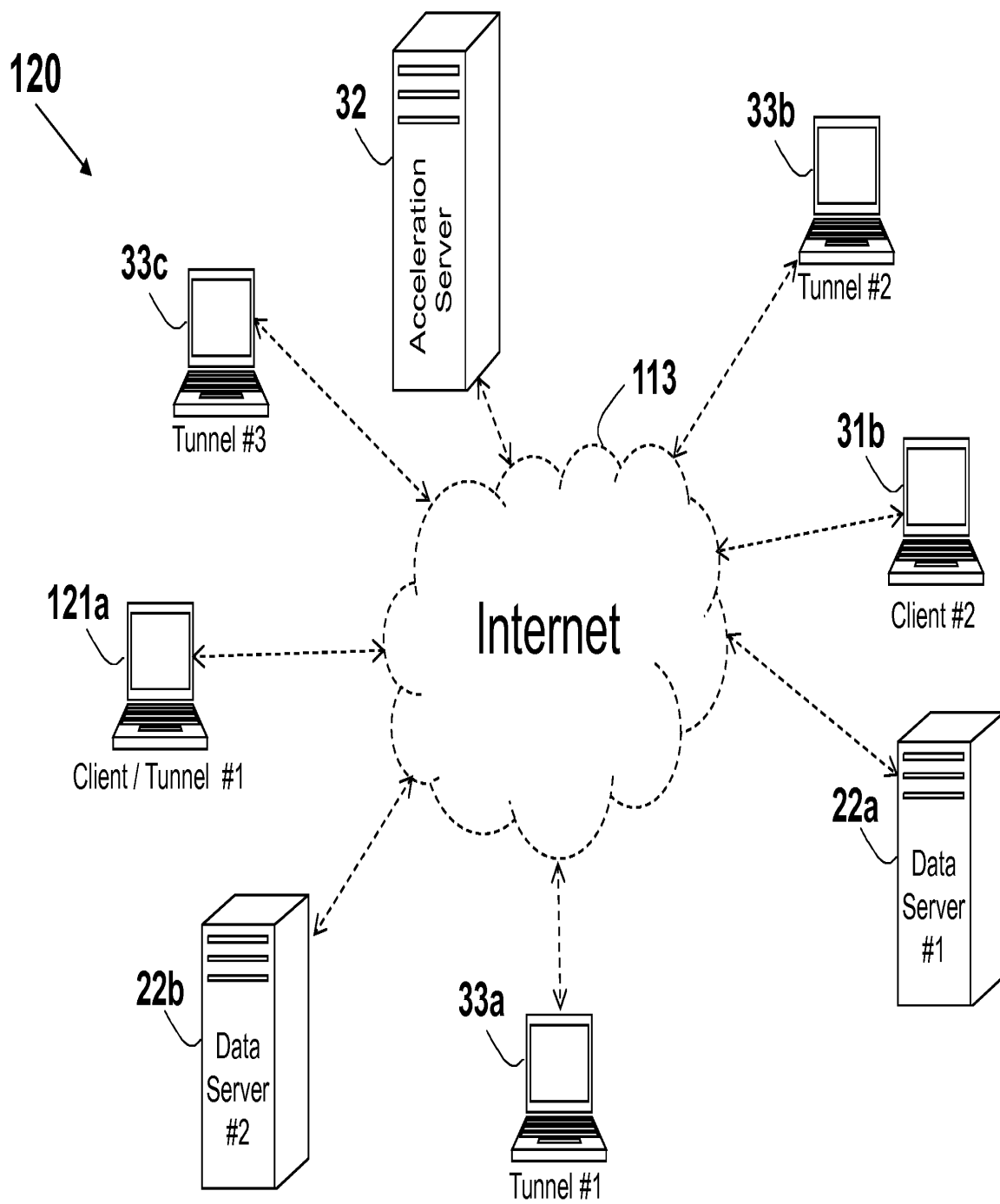


FIG. 12

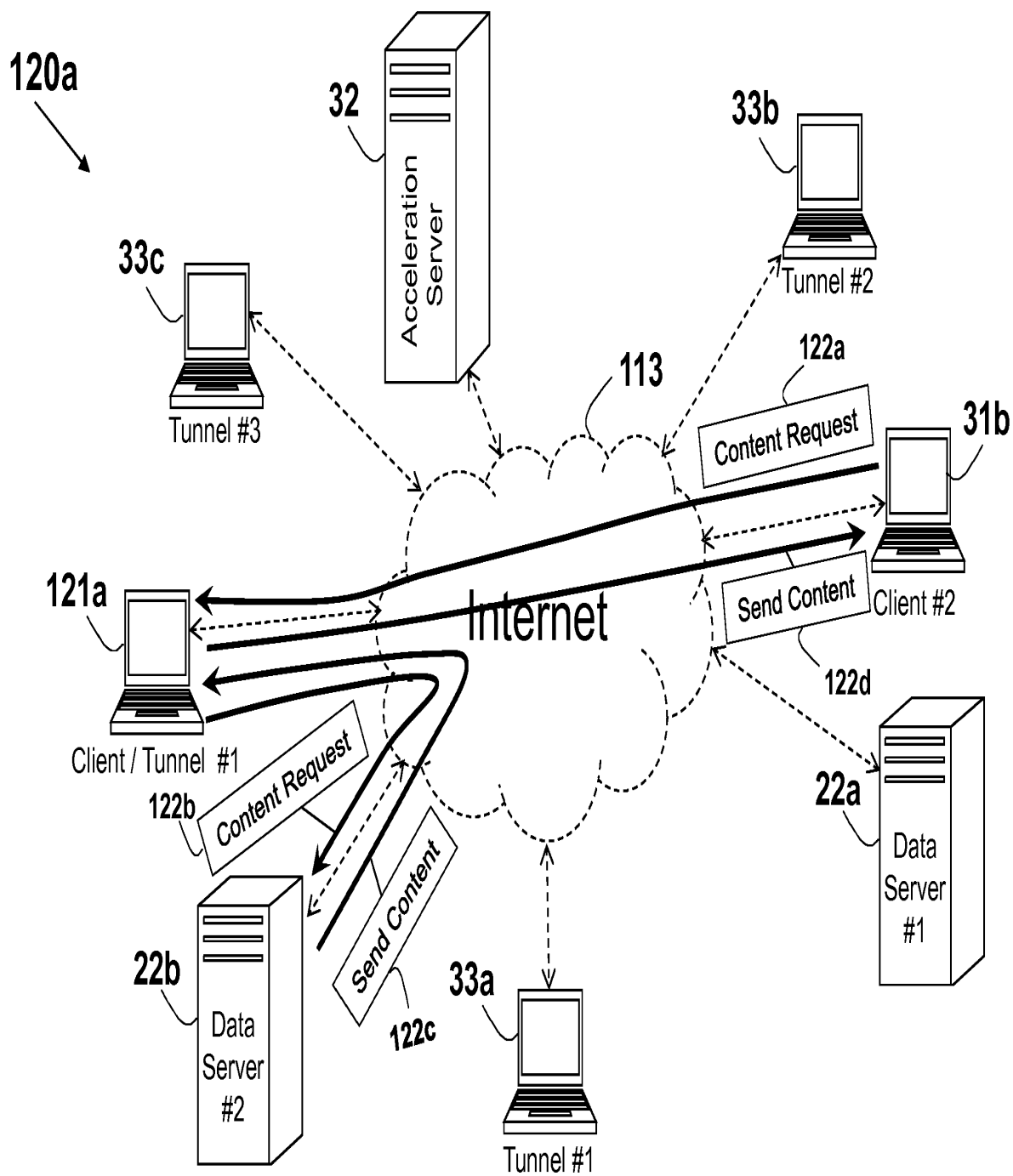


FIG. 12a

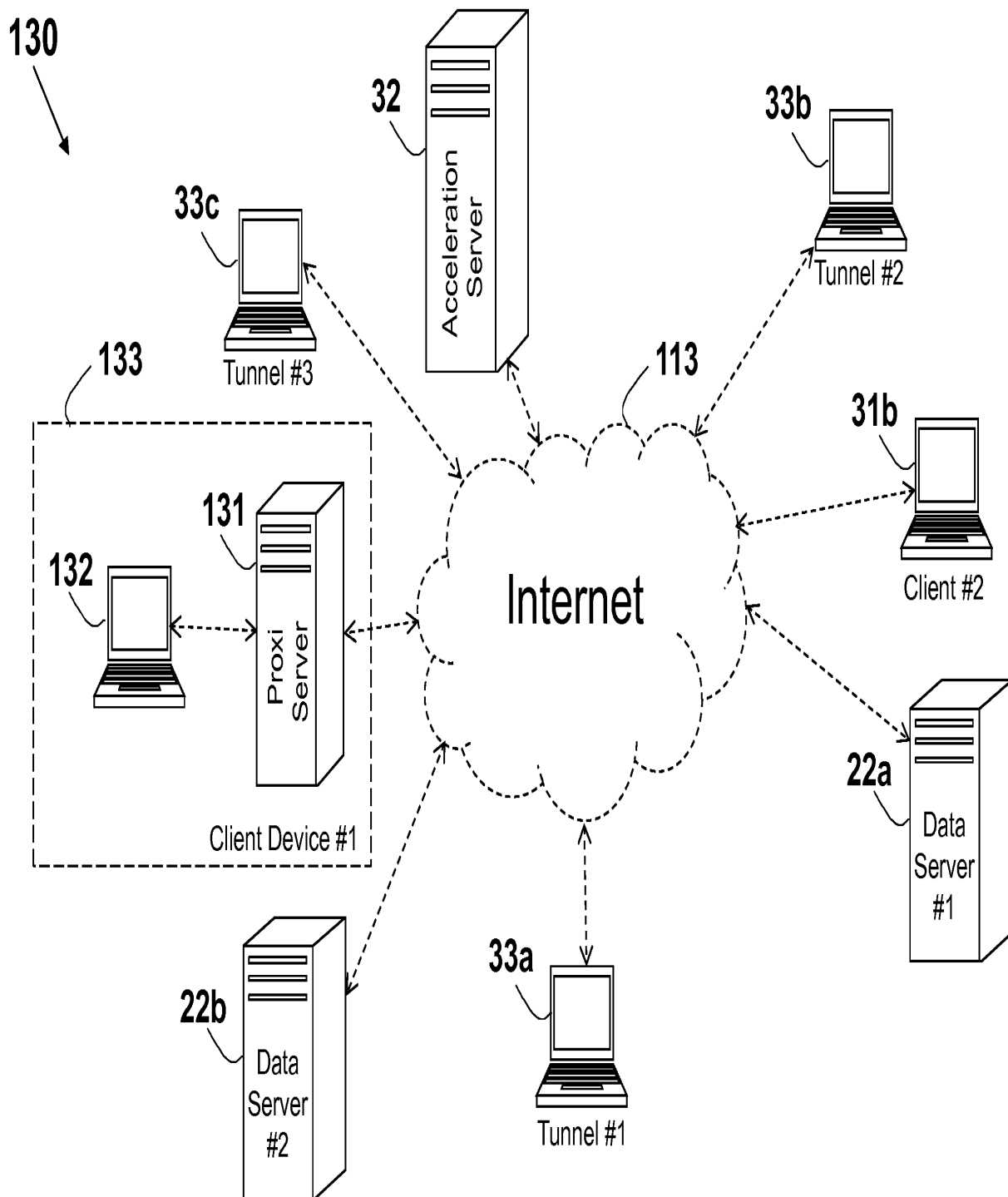


FIG. 13

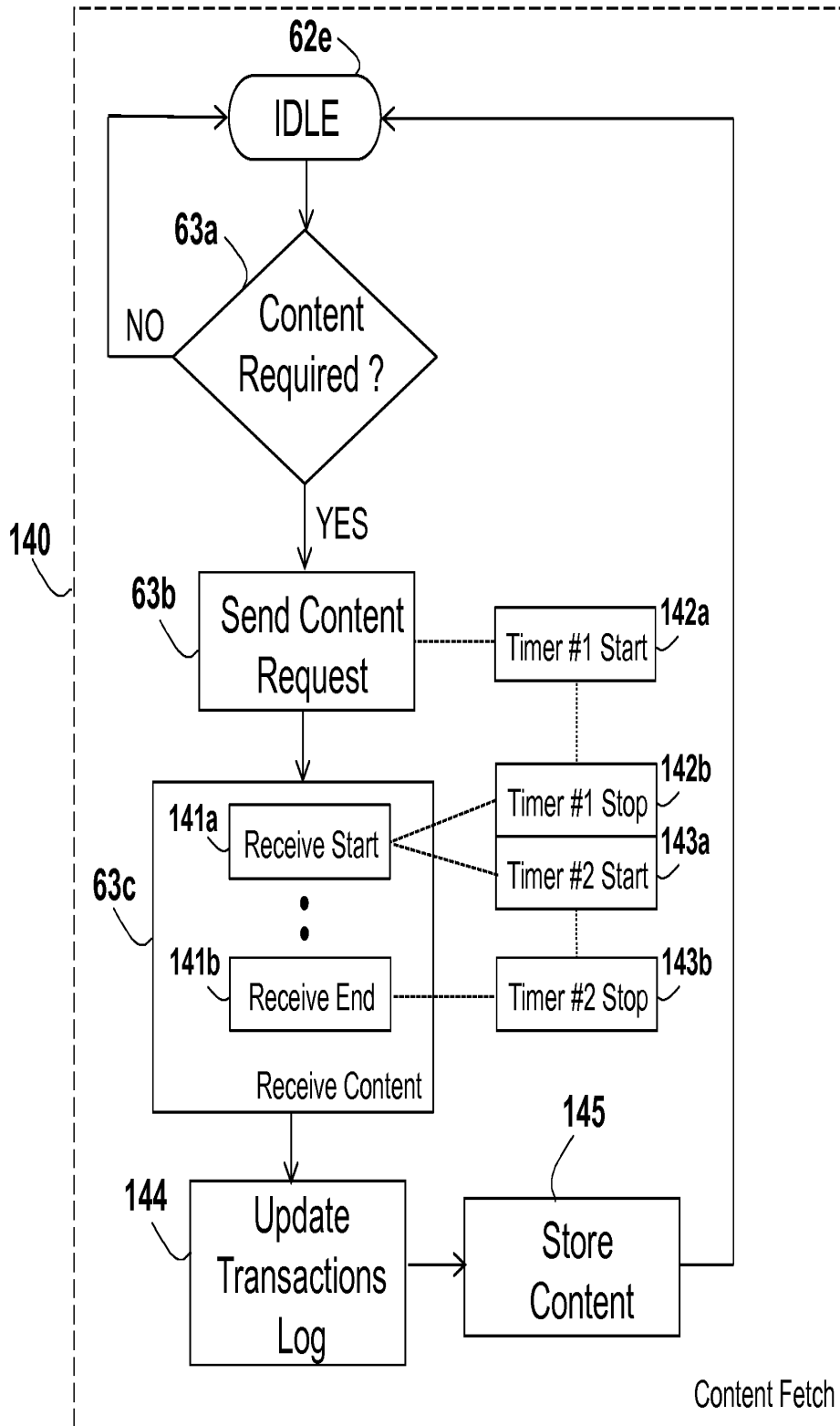


FIG. 14

150



151a



151b



151c



151d



151e



151f



152 →

152a →

152b →

152c →

152d →

DATE / TIME	TUNNEL IP ADDRESS	SERVER IP ADDRESS	URL	BW (Kb/s)	RTT (ms)
13/3 9:23	229.155.81.168	128.164.35.142	www.111.com/22.mpg	1000	30
13/3 9:46	248.107.109.10	49.154.2.5	www.xxx.com/hy.avi	350	70
15/4 11:22	158.217.19.195	72.251.238.51	www.yyy.com/t6.php	2500	540
16/4 16:43	253.6.96.135	169.101.27.217	www.1er.com/5.shtml	1400	170

FIG. 15

150a



153a



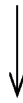
153b



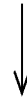
153c



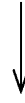
153d



153e



153f



154 →

154a →

154b →

154c →

154d →

DATE / TIME	TUNNEL IP ADDRESS	SERVER IP ADDRESS	URL	BW (Kb/s)	RTT (ms)
13/3 9:23	139.230.154.213	49.154.2.5	https://www.youtube.com/ watch?v=cJwNwCD6CAE	1000	30
13/3 9:46	132.171.60.197	49.154.2.5	https://www.youtube.com/ watch?v=FRAbW6oCgHM	350	70
15/4 11:22	248.46.80.36	49.154.2.5	https://www.youtube.com/ watch?v=CQQIEQh8pU0	2500	540
16/4 16:43	31.16.208.171	49.154.2.5	https://www.youtube.com/ watch?v=KJbtX1xWyMA	1400	170

FIG. 15a

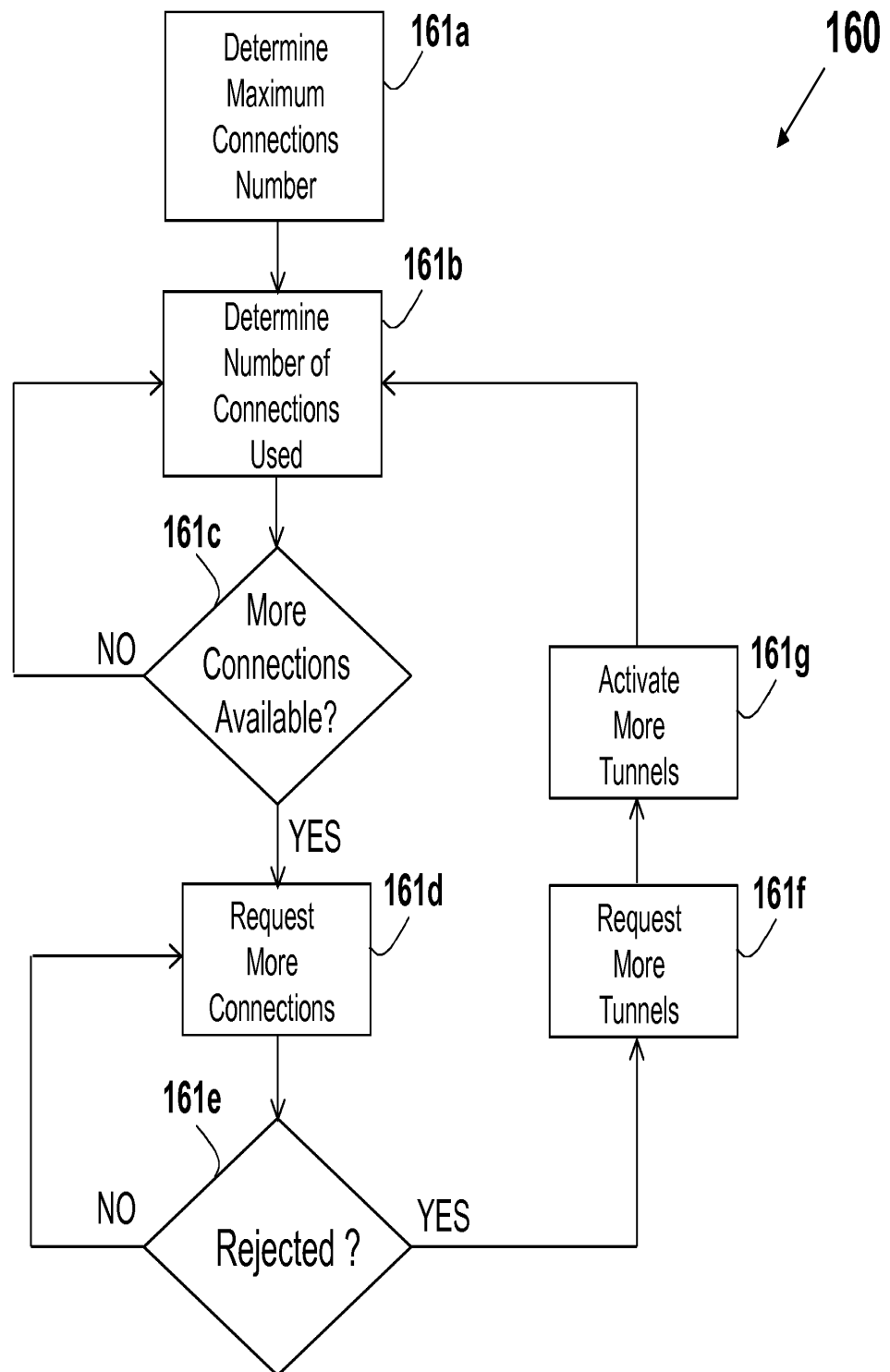


FIG. 16

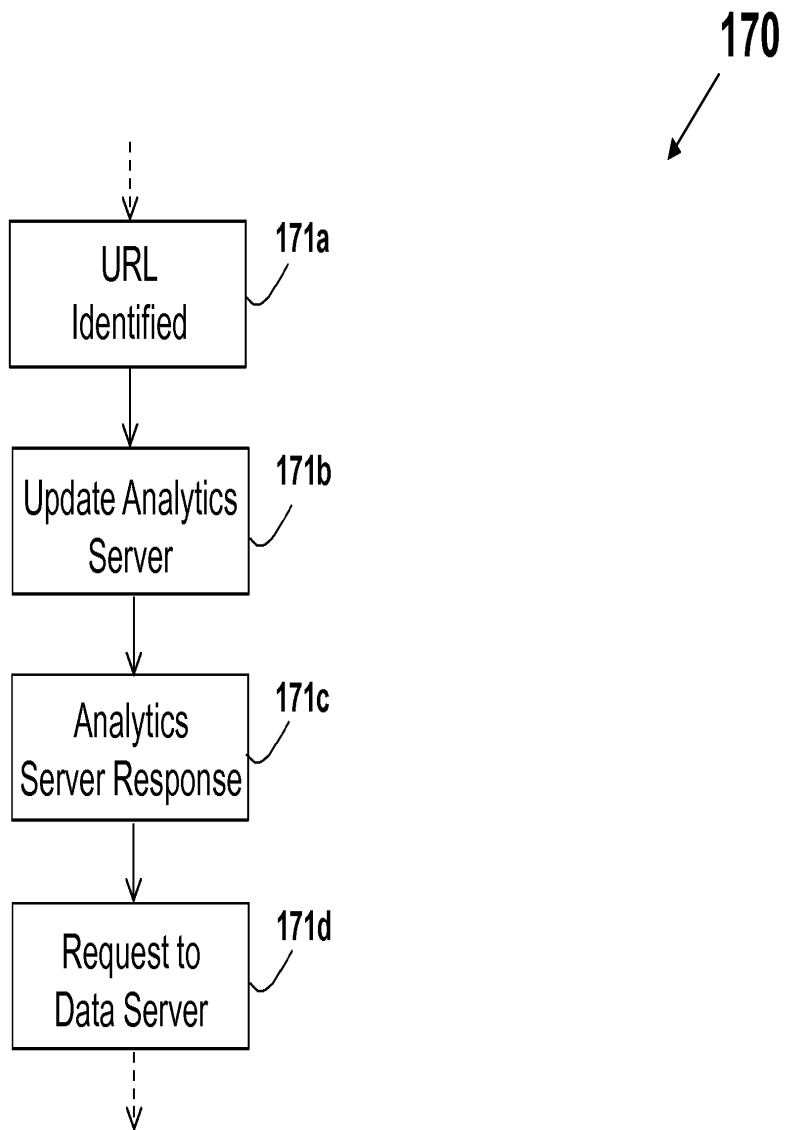


FIG. 17 (Prior Art)

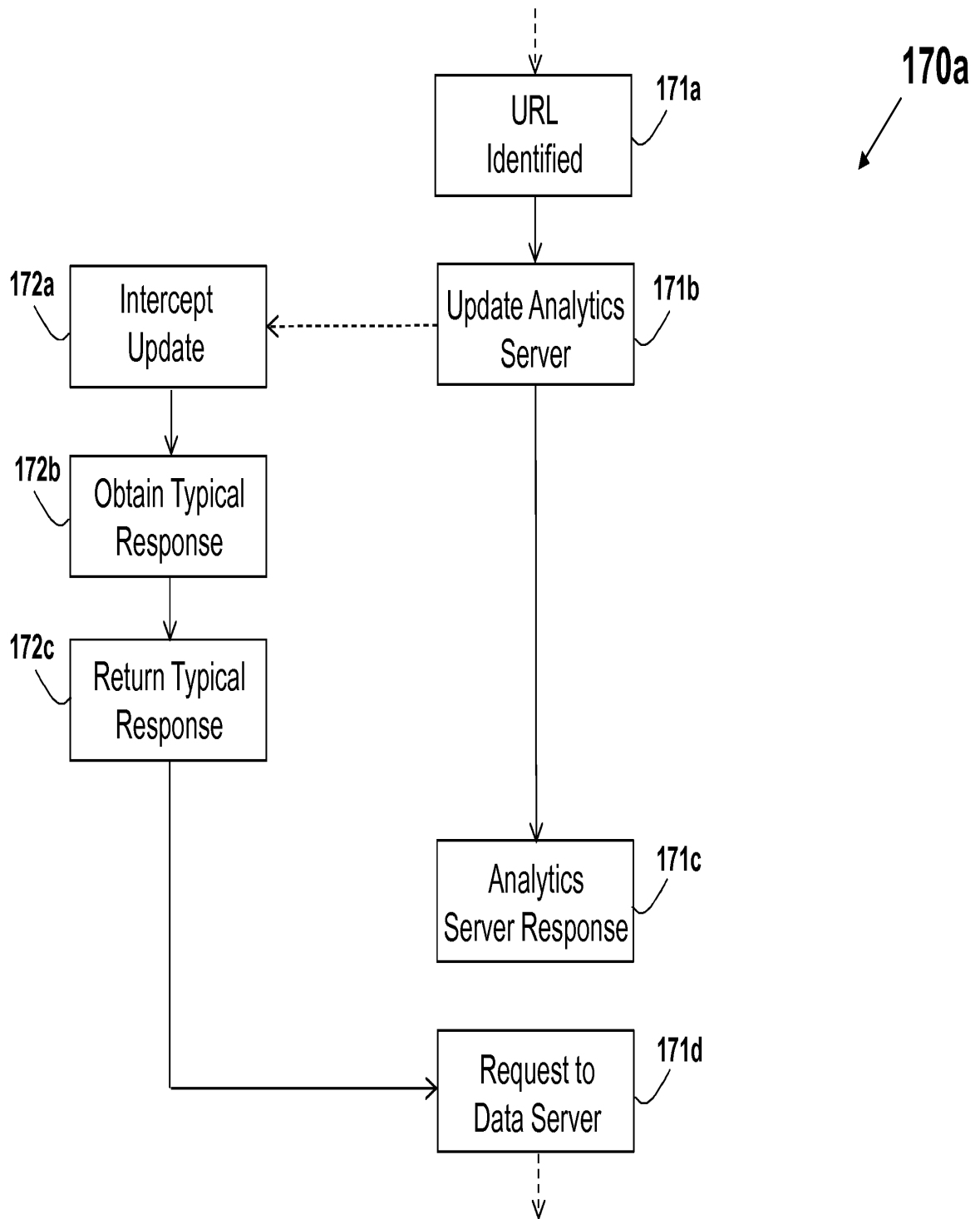


FIG. 17a

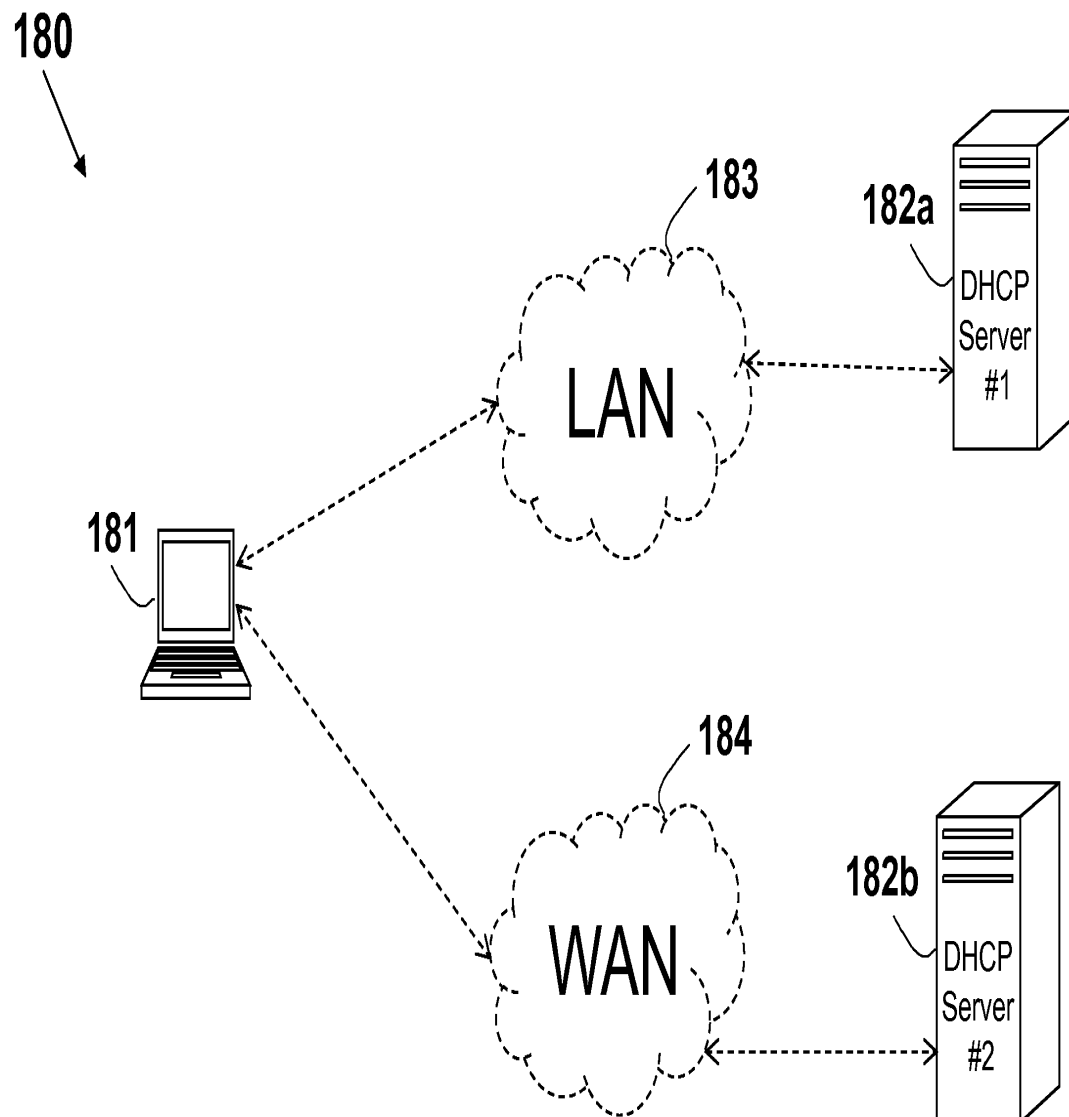


FIG. 18 (Prior Art)

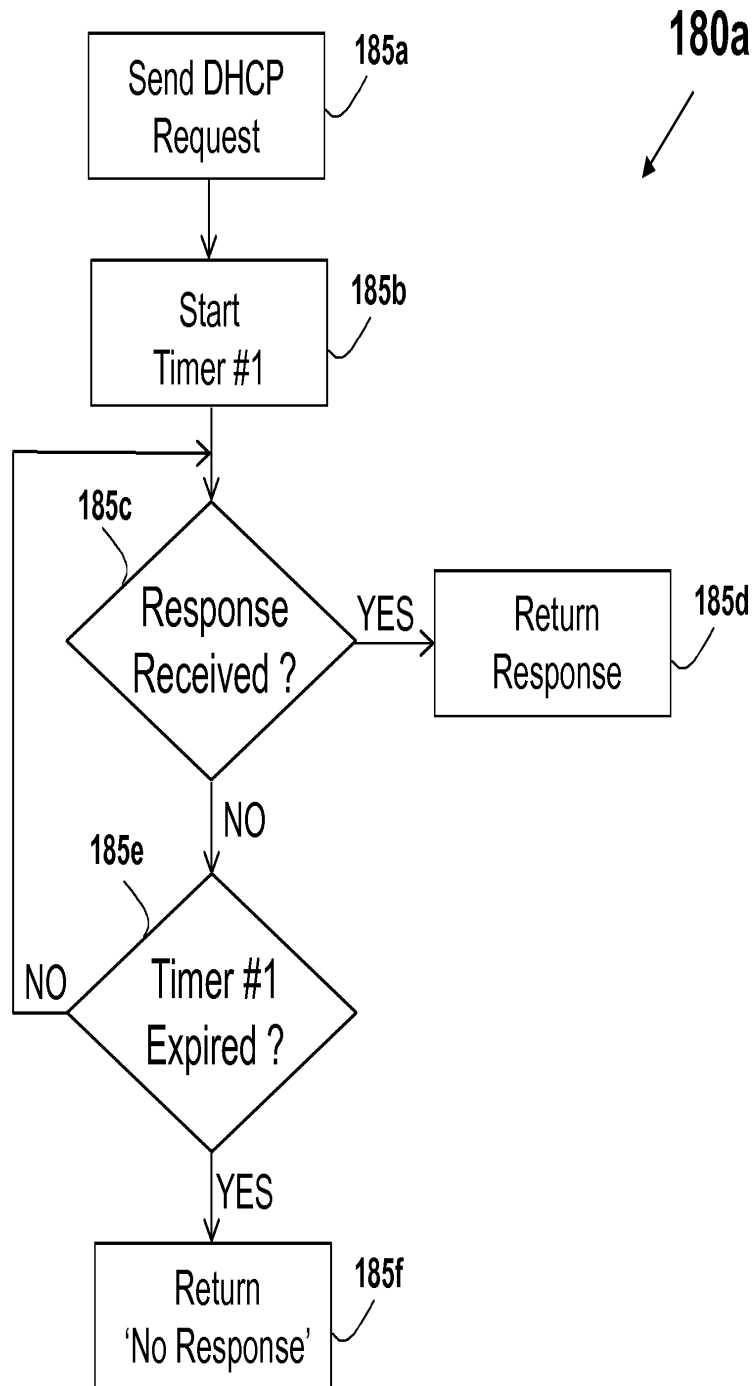


FIG. 18a (Prior Art)

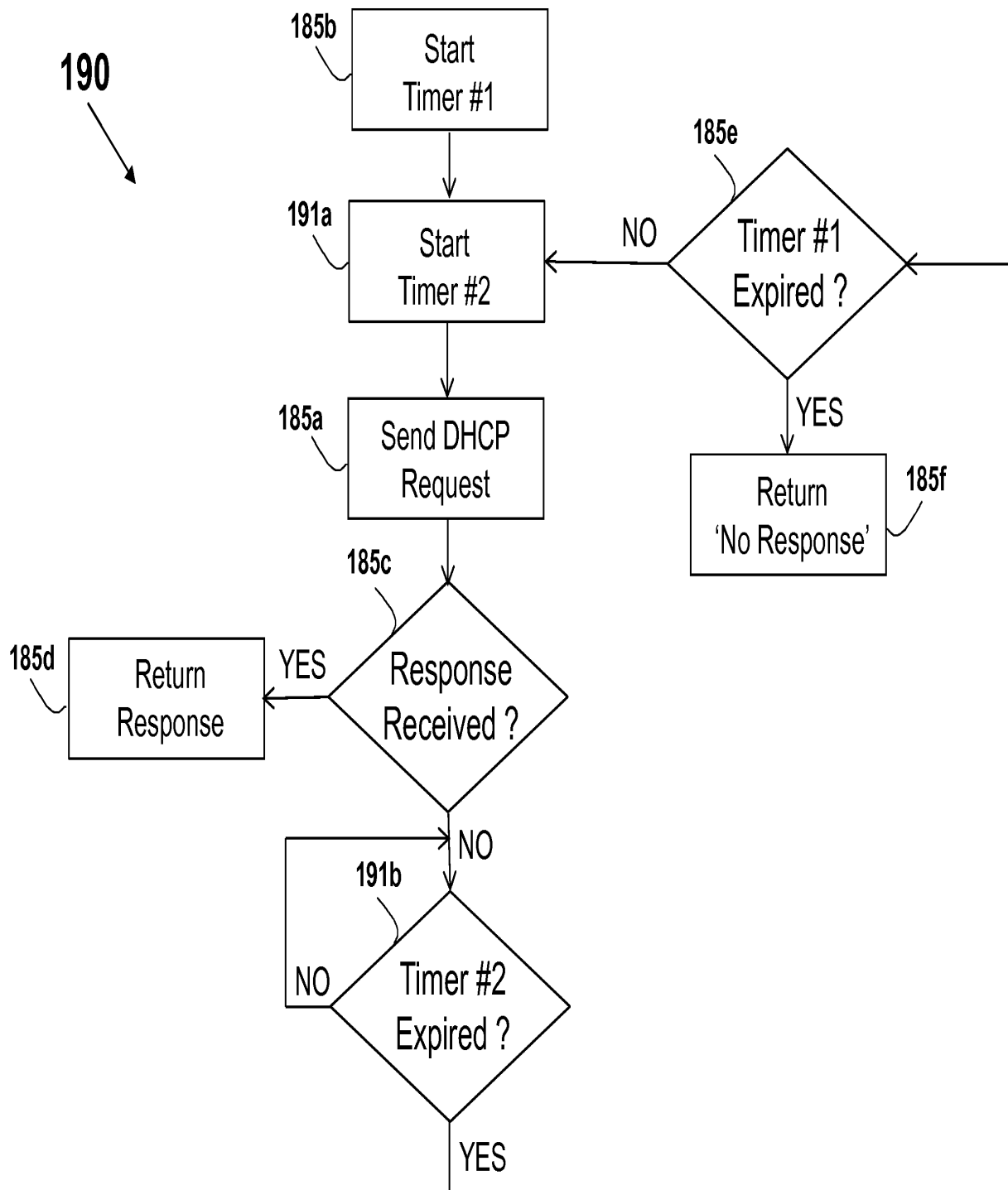


FIG. 19

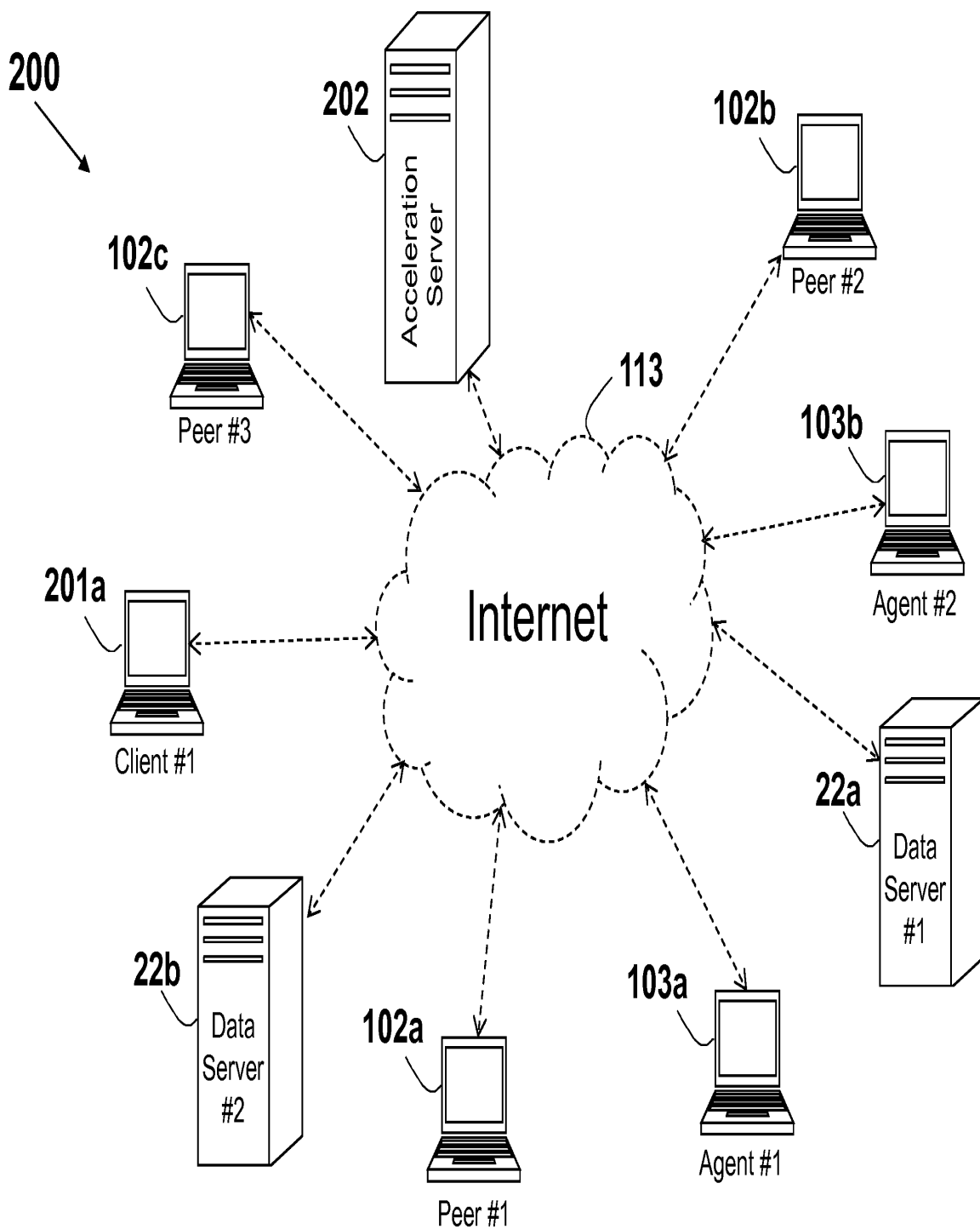


FIG. 20

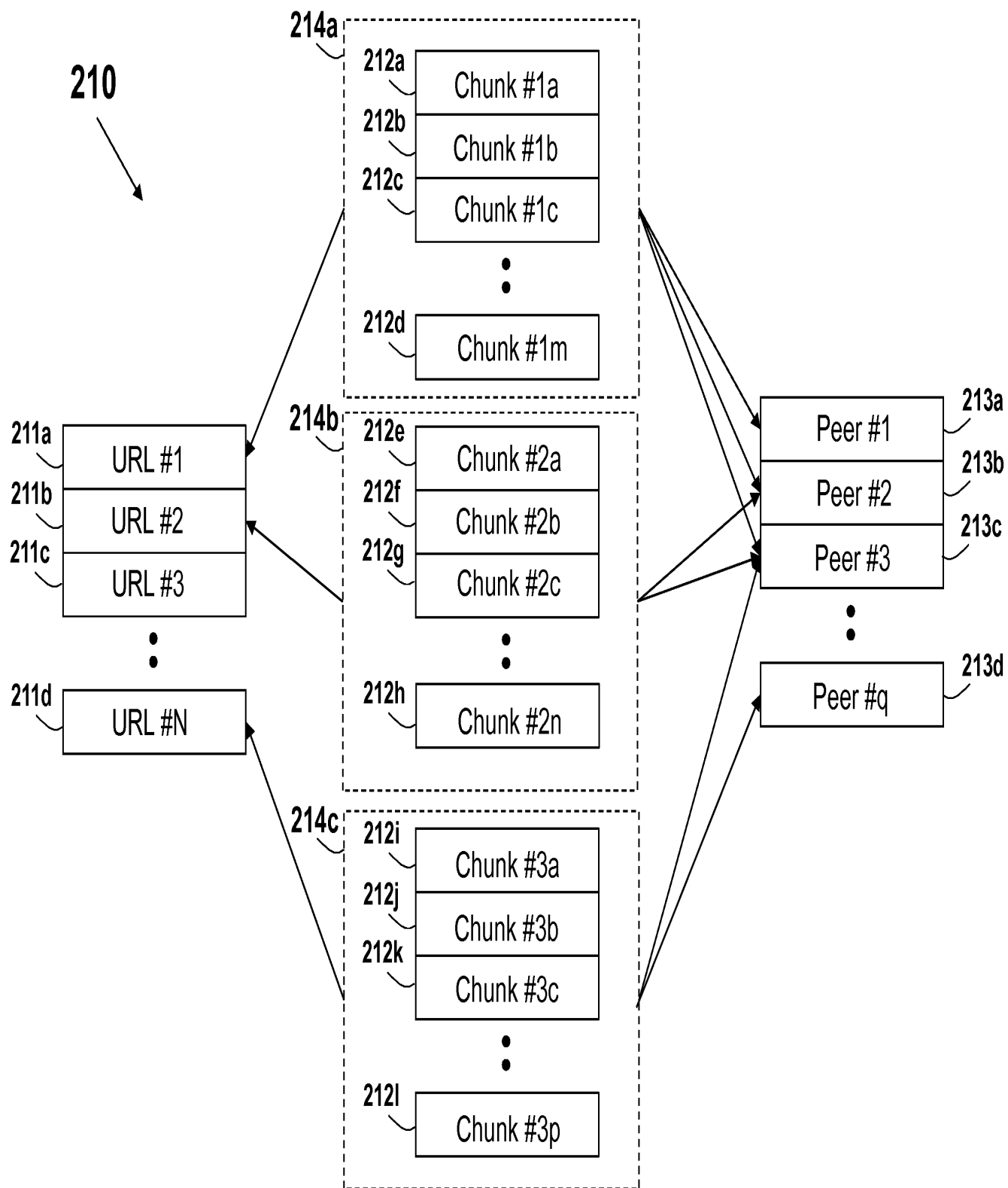


FIG. 21

210a

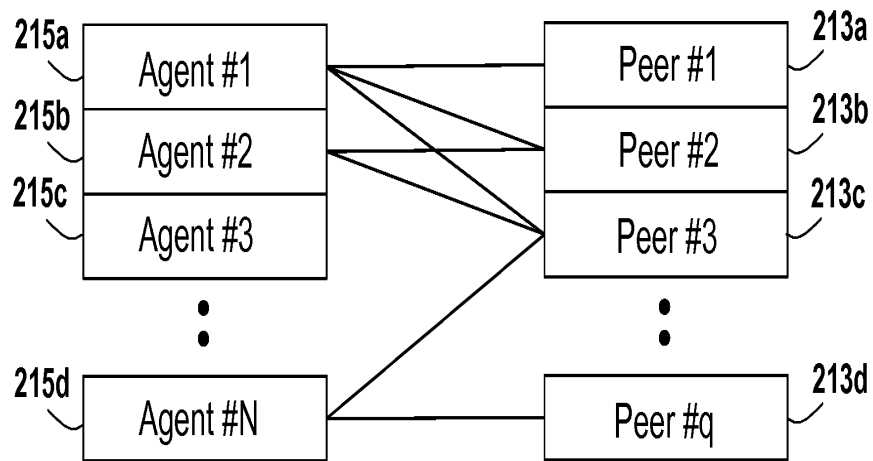


FIG. 21a

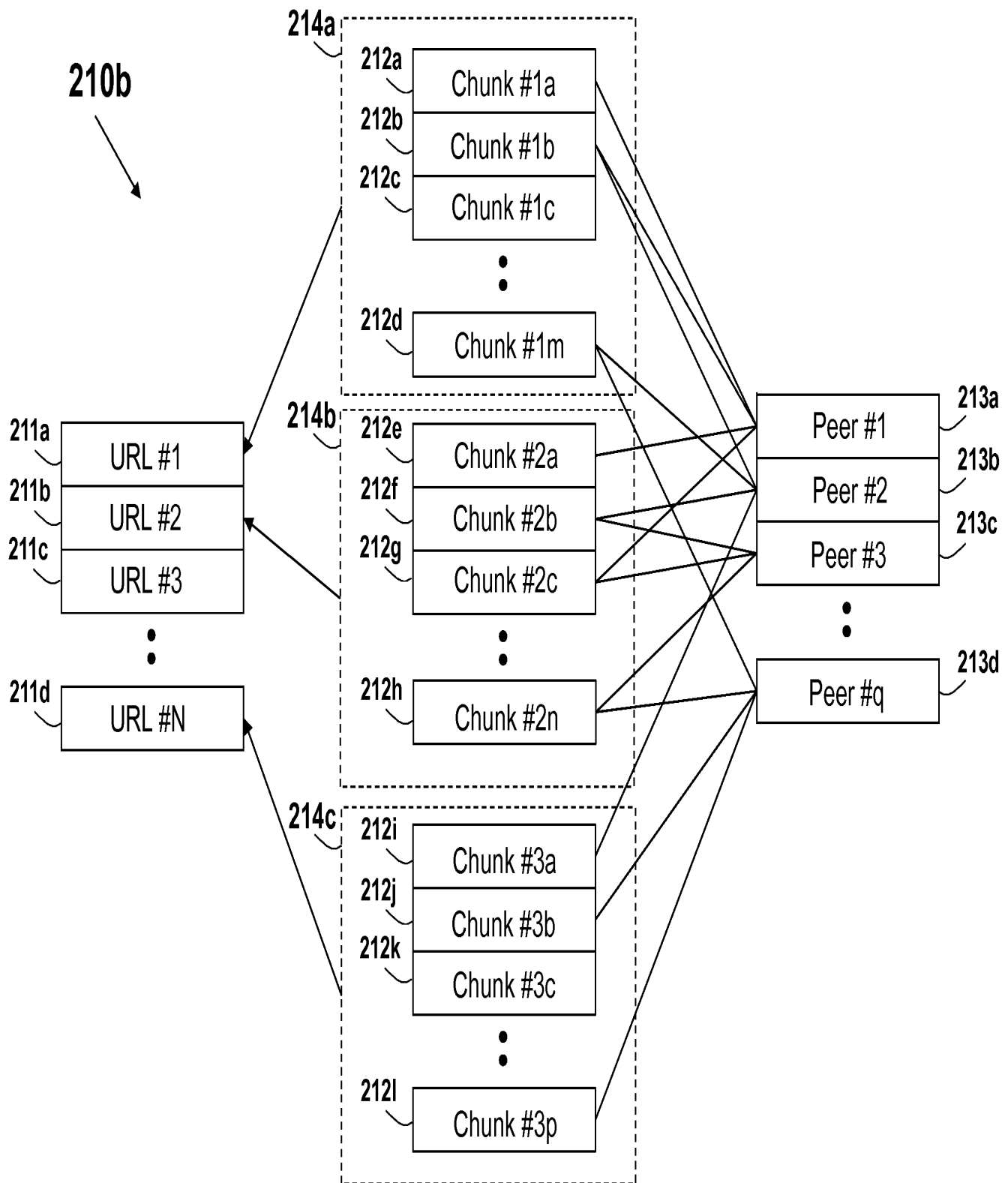


FIG. 21b

210c

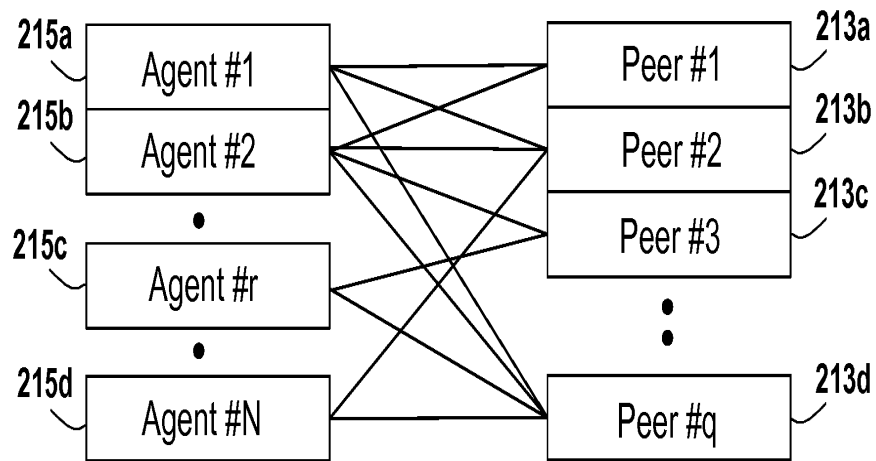


FIG. 21c

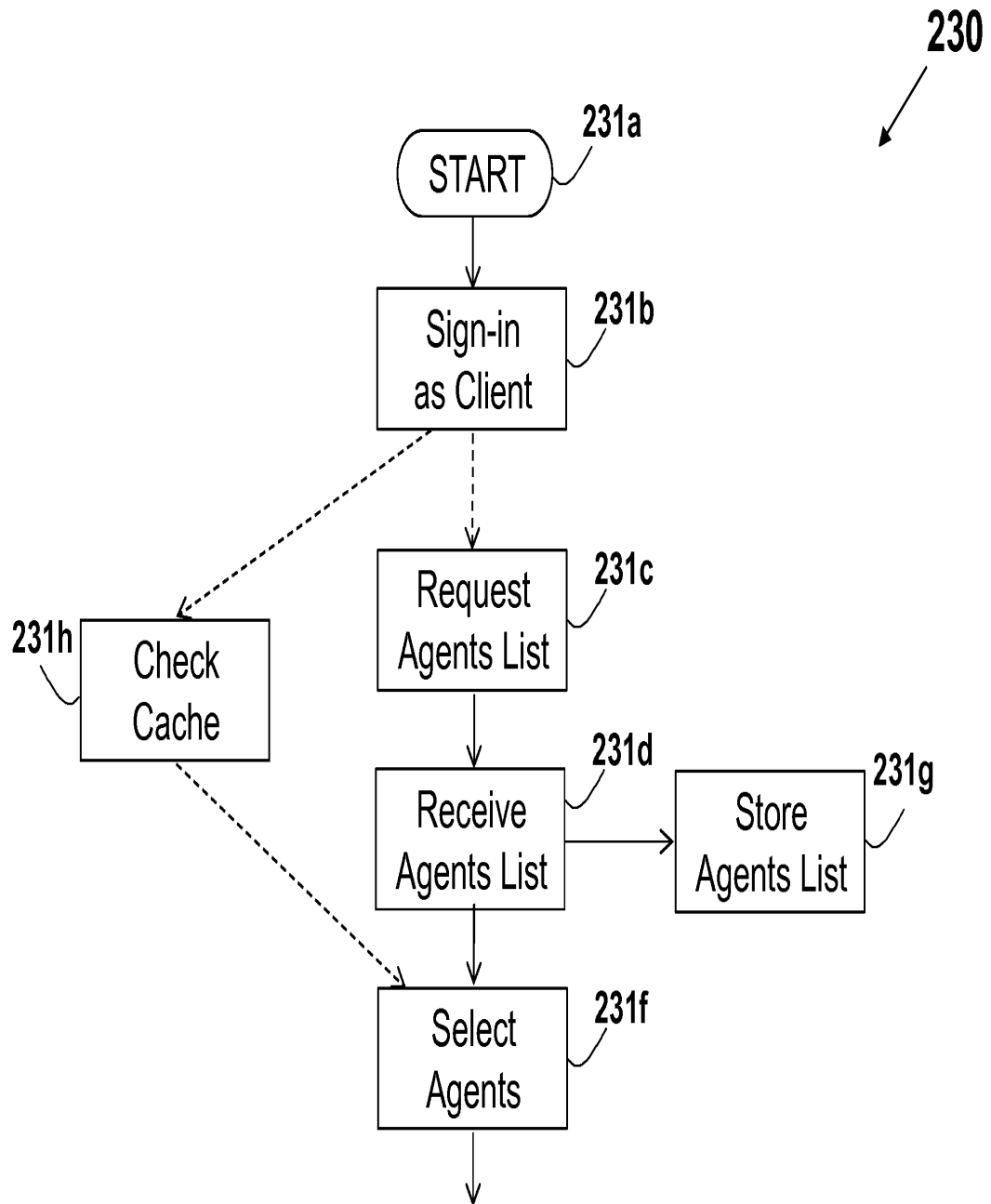


FIG. 23

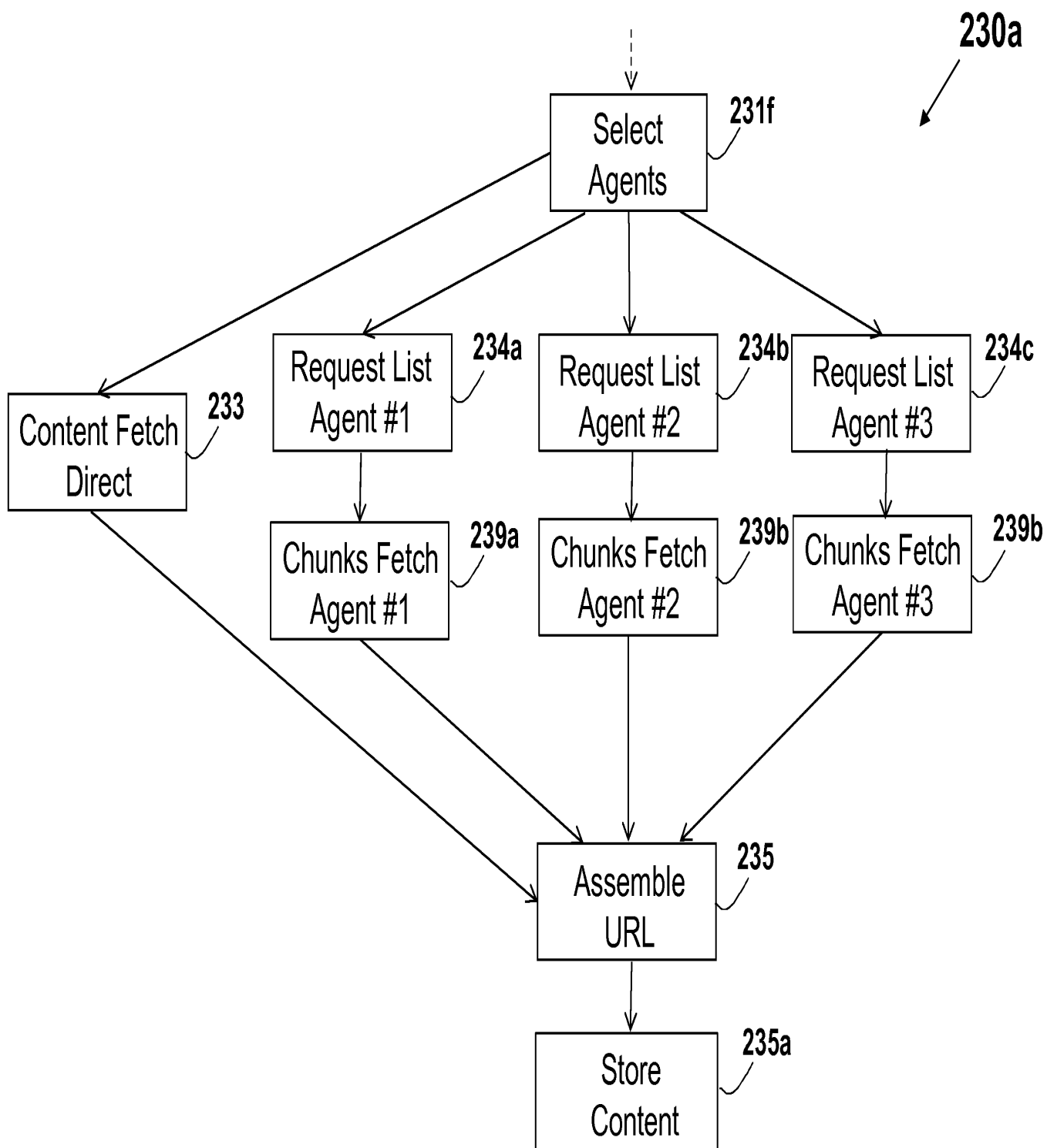


FIG. 23a

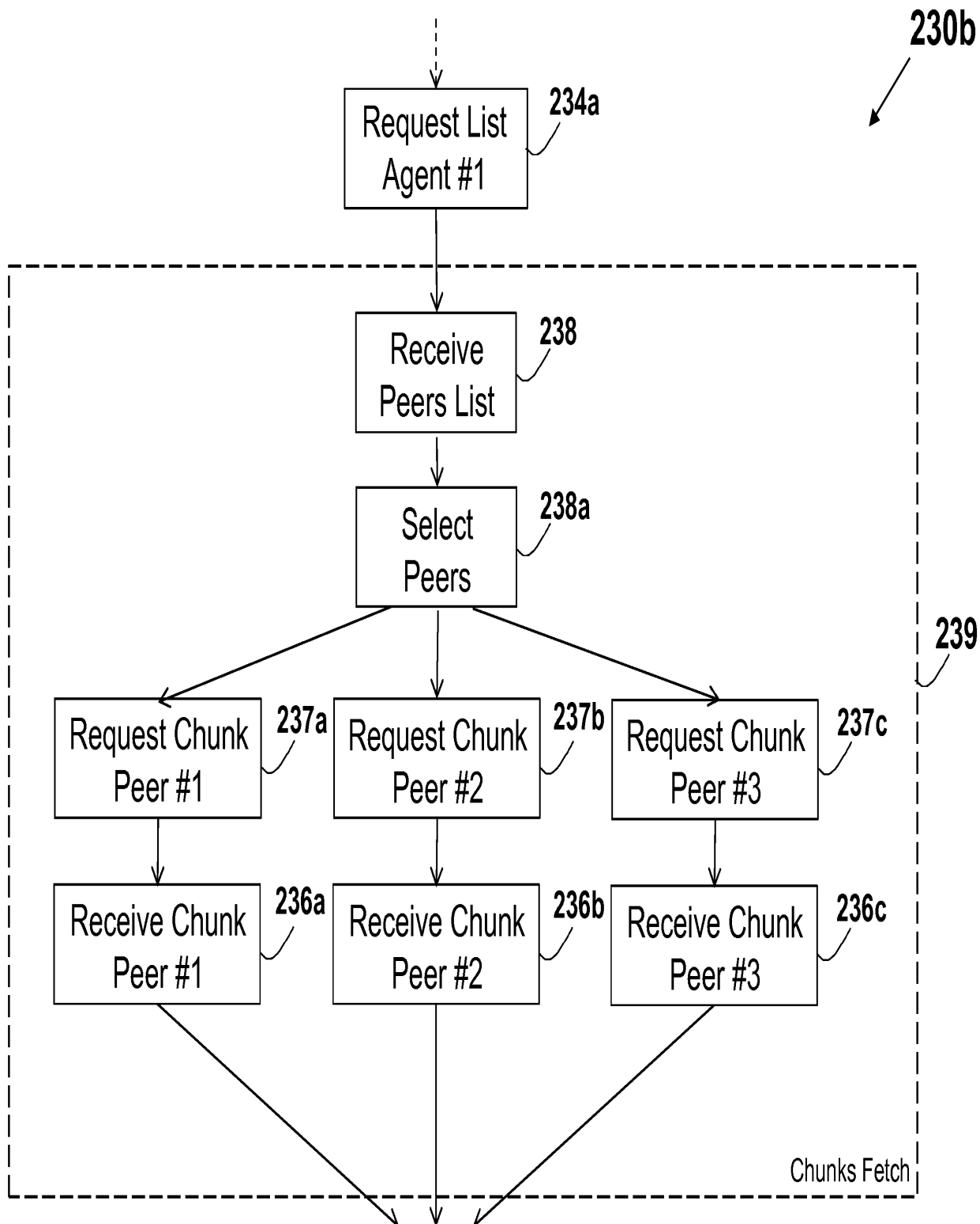


FIG. 23b

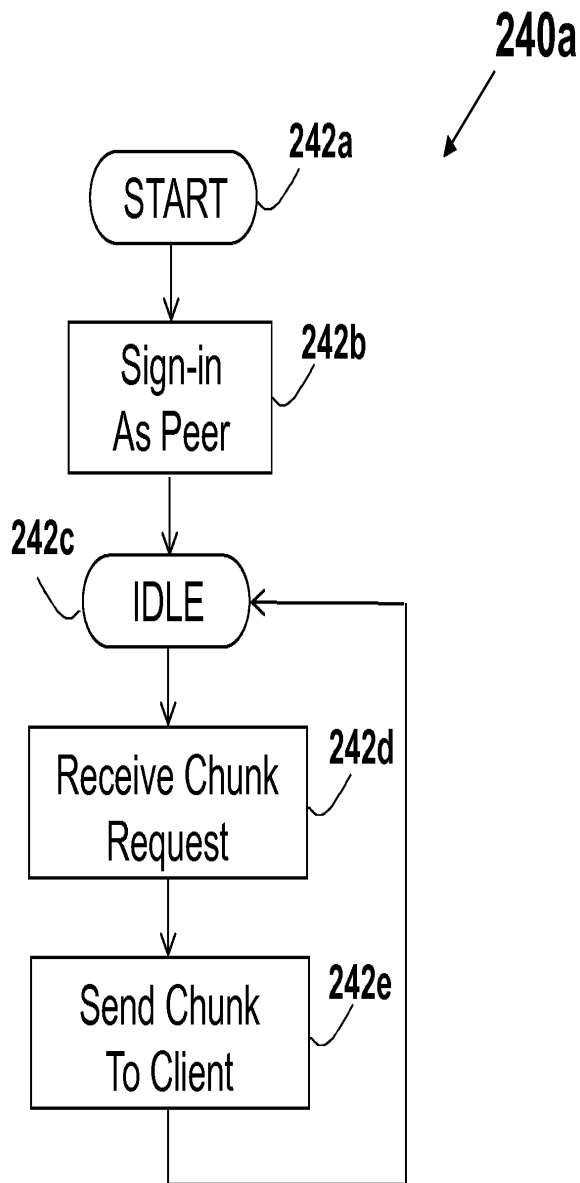


FIG. 24a

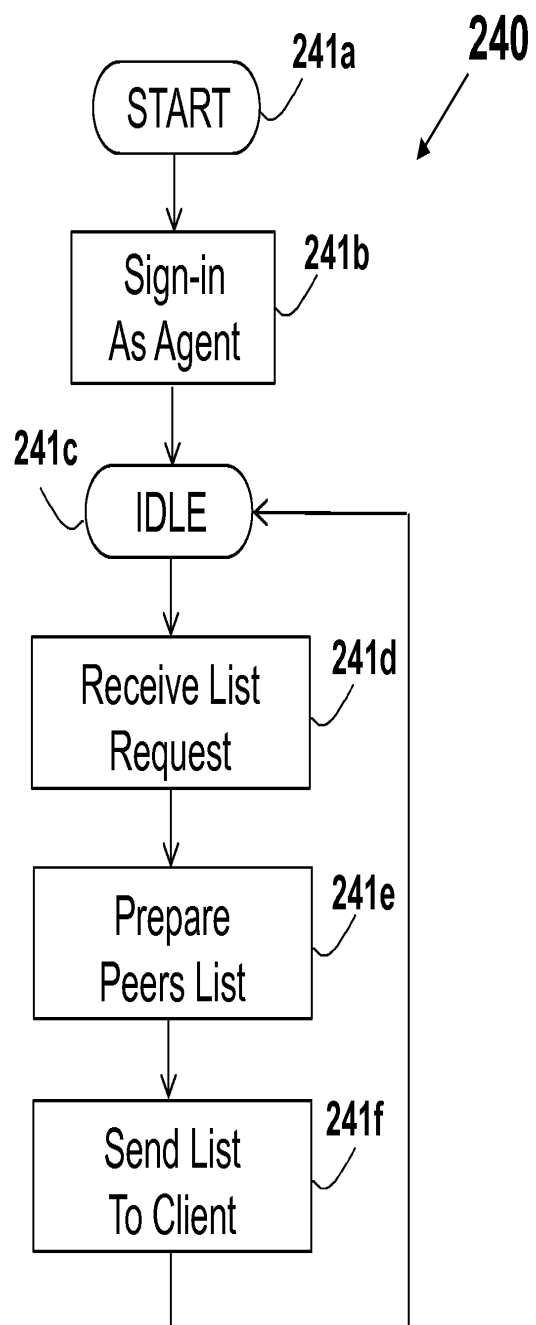


FIG. 24

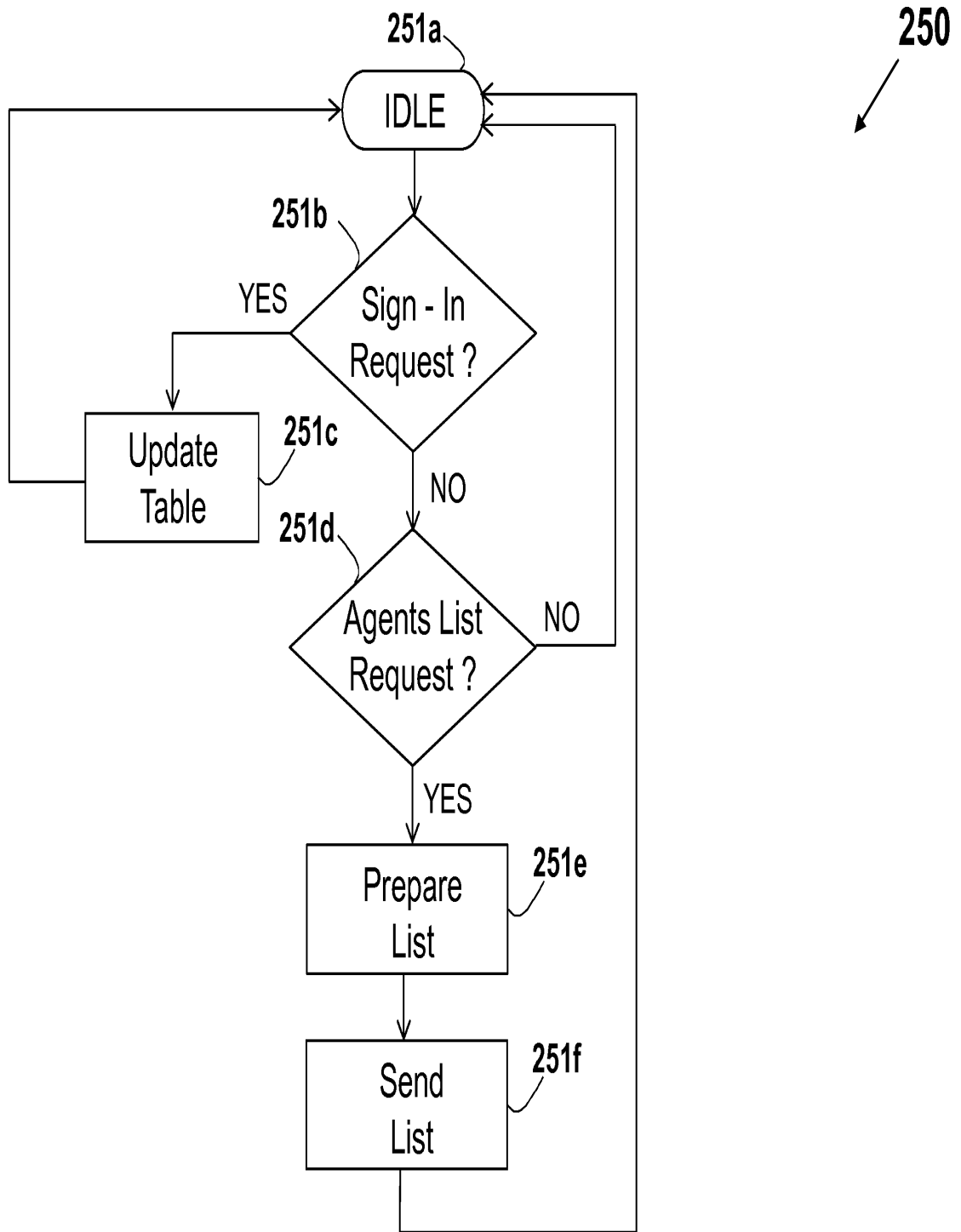


FIG. 25

250a



252a



252b



252c



252d



253 →

TYPE

IP ADDRESS

SIGN-IN
DATE / TIME

LOCATION

253a →

Agent

73.0.82.8

24/3 8:35

Miami, FL, USA

253b →

Agent

68.78.78.3

25/3 10:59

Berlin, Germany

253c →

Peer

111.13.69.78

28/3 11:49

Melbourne, Australia

253d →

Client

125.90.25.92

29/3 14:23

Jerusalem, Israel

253e →

Agent / Peer

95.33.37.80

16/3 21:53

Beijing, China

FIG. 25a

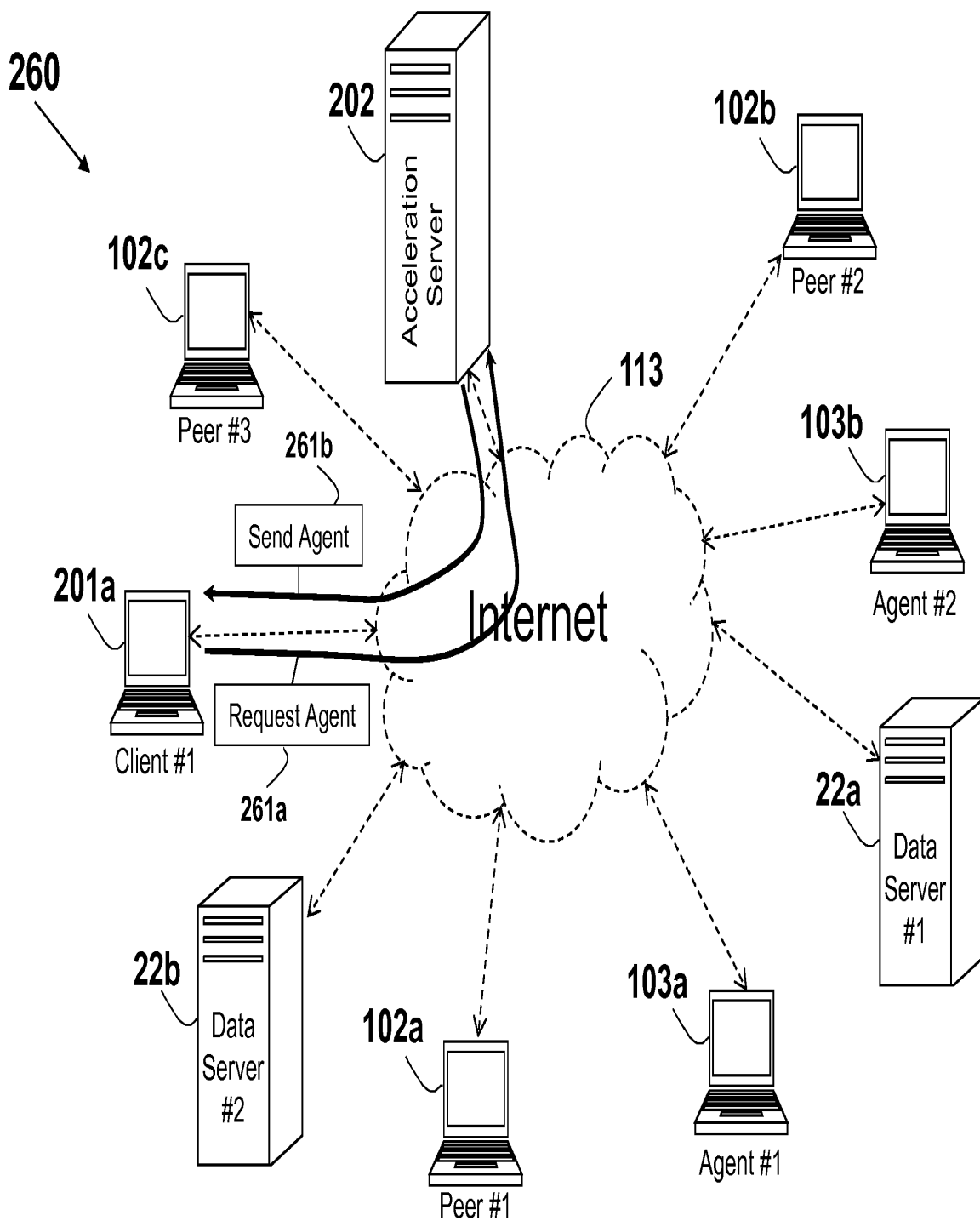


FIG. 26

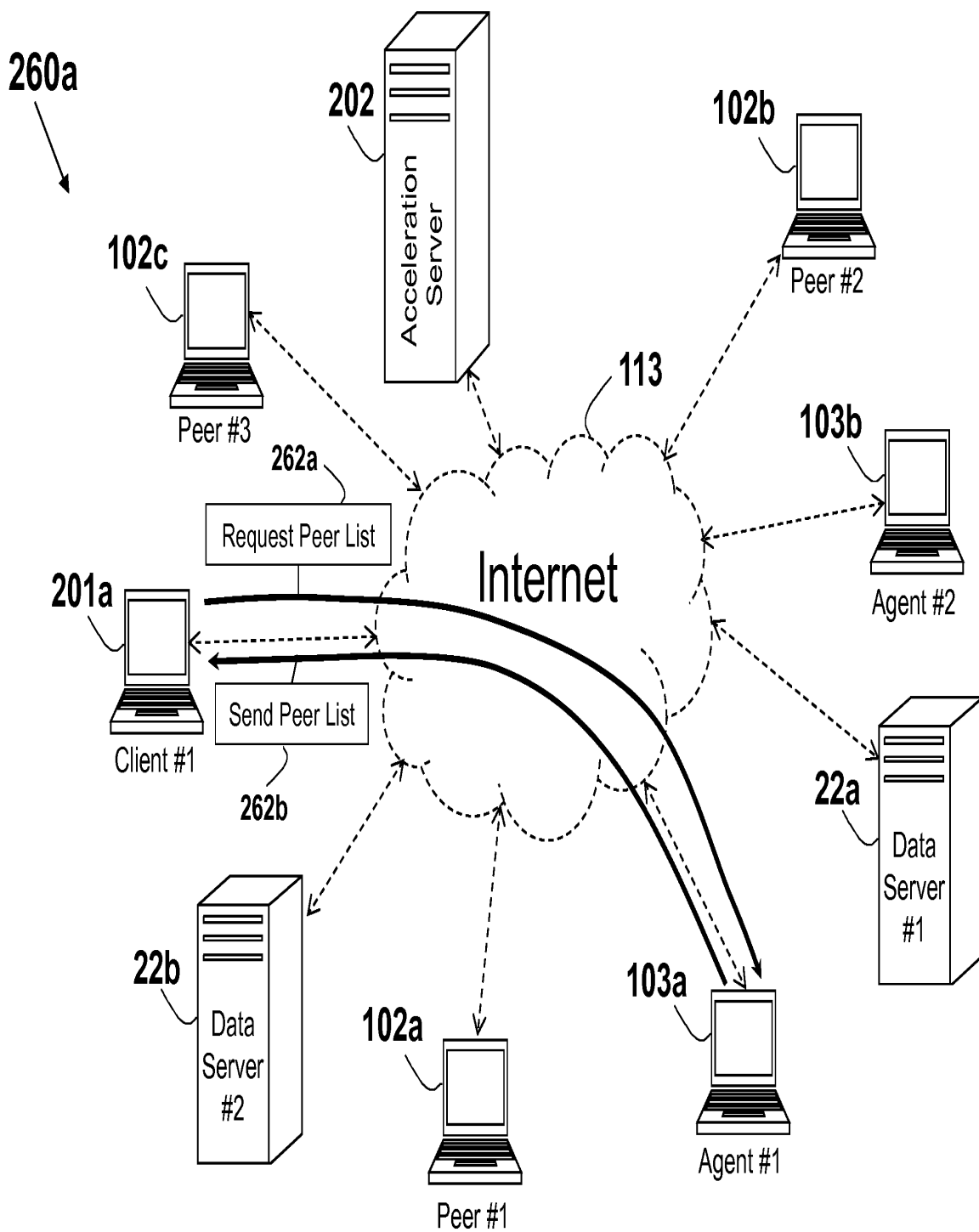


FIG. 26a

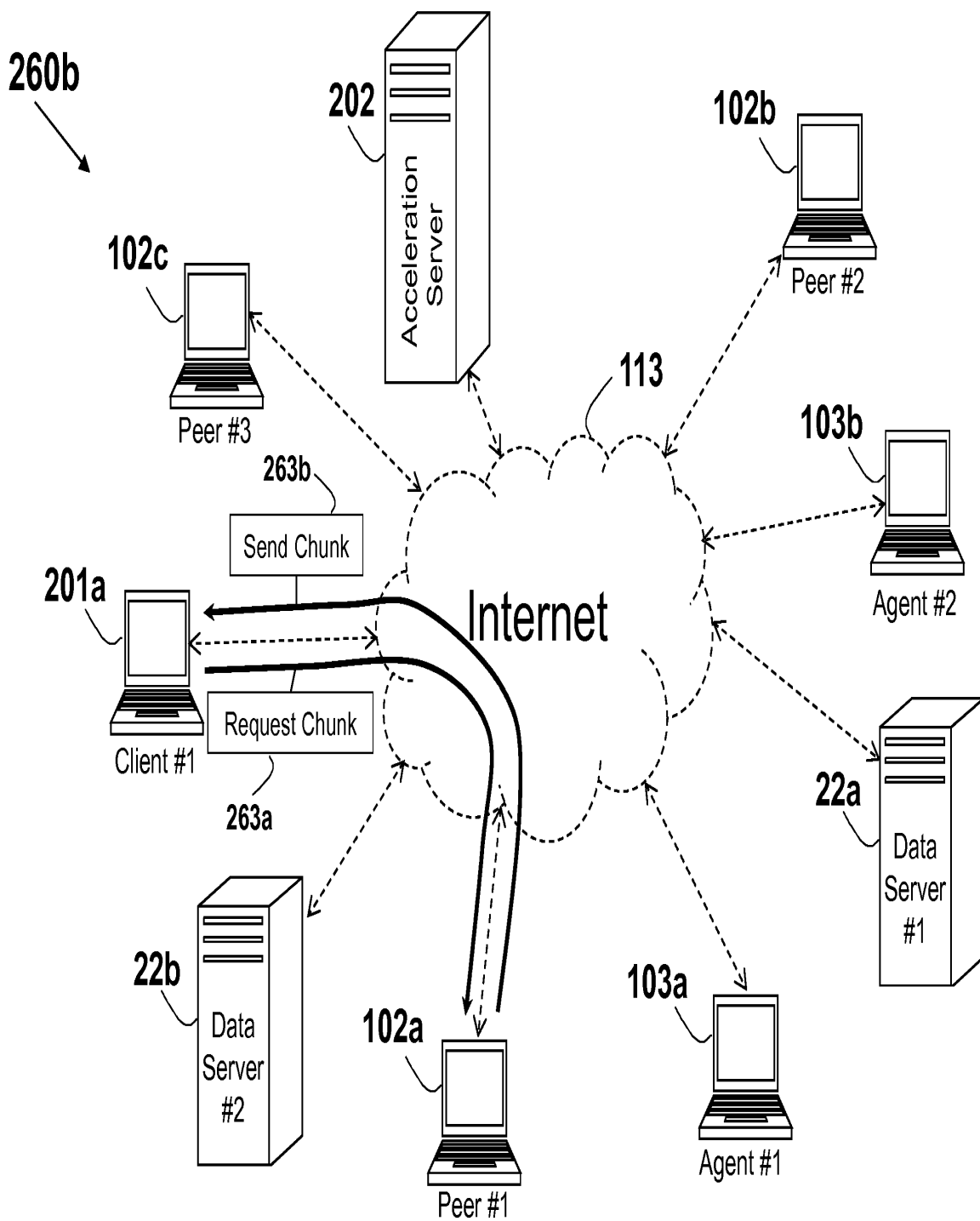


FIG. 26b

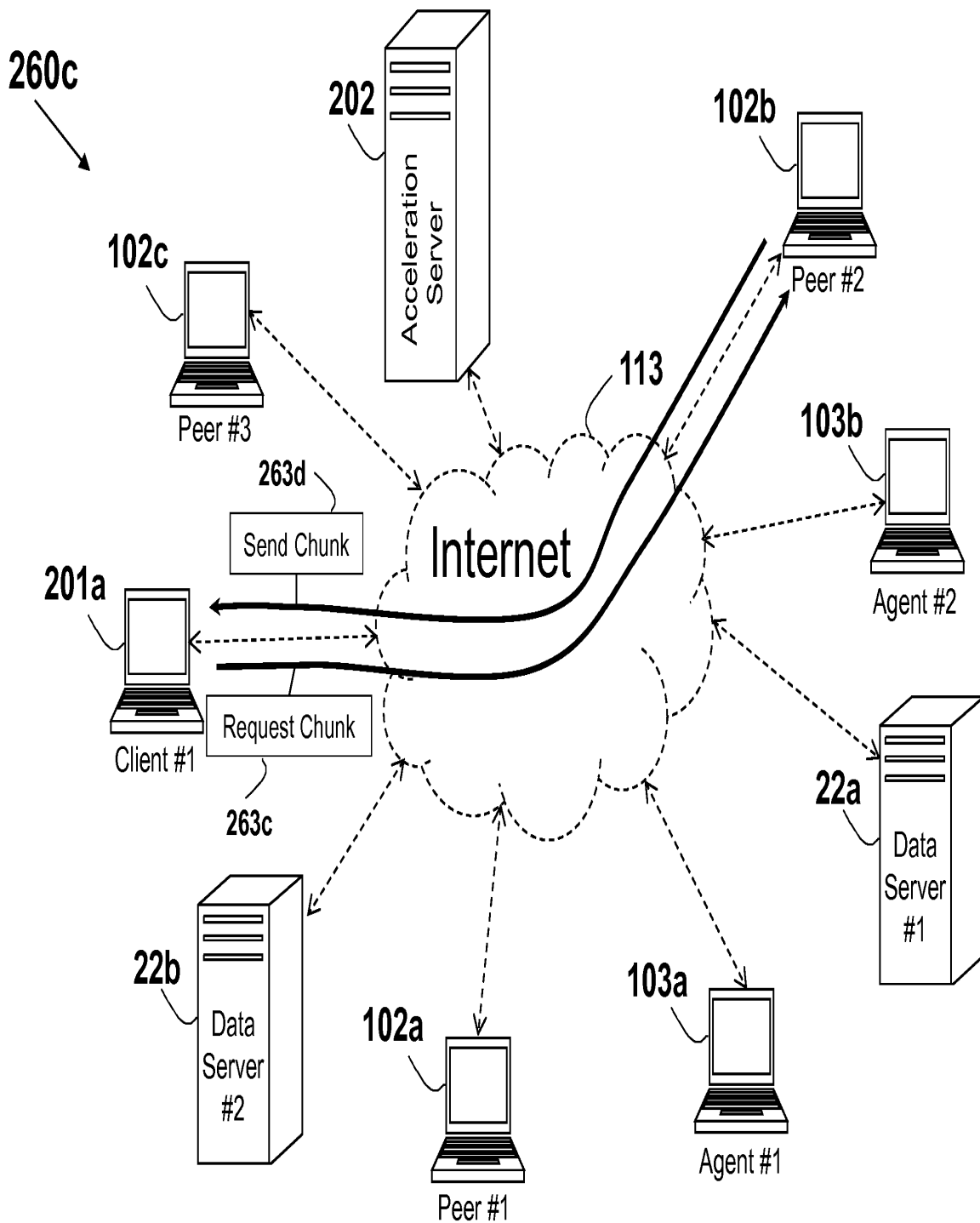


FIG. 26c

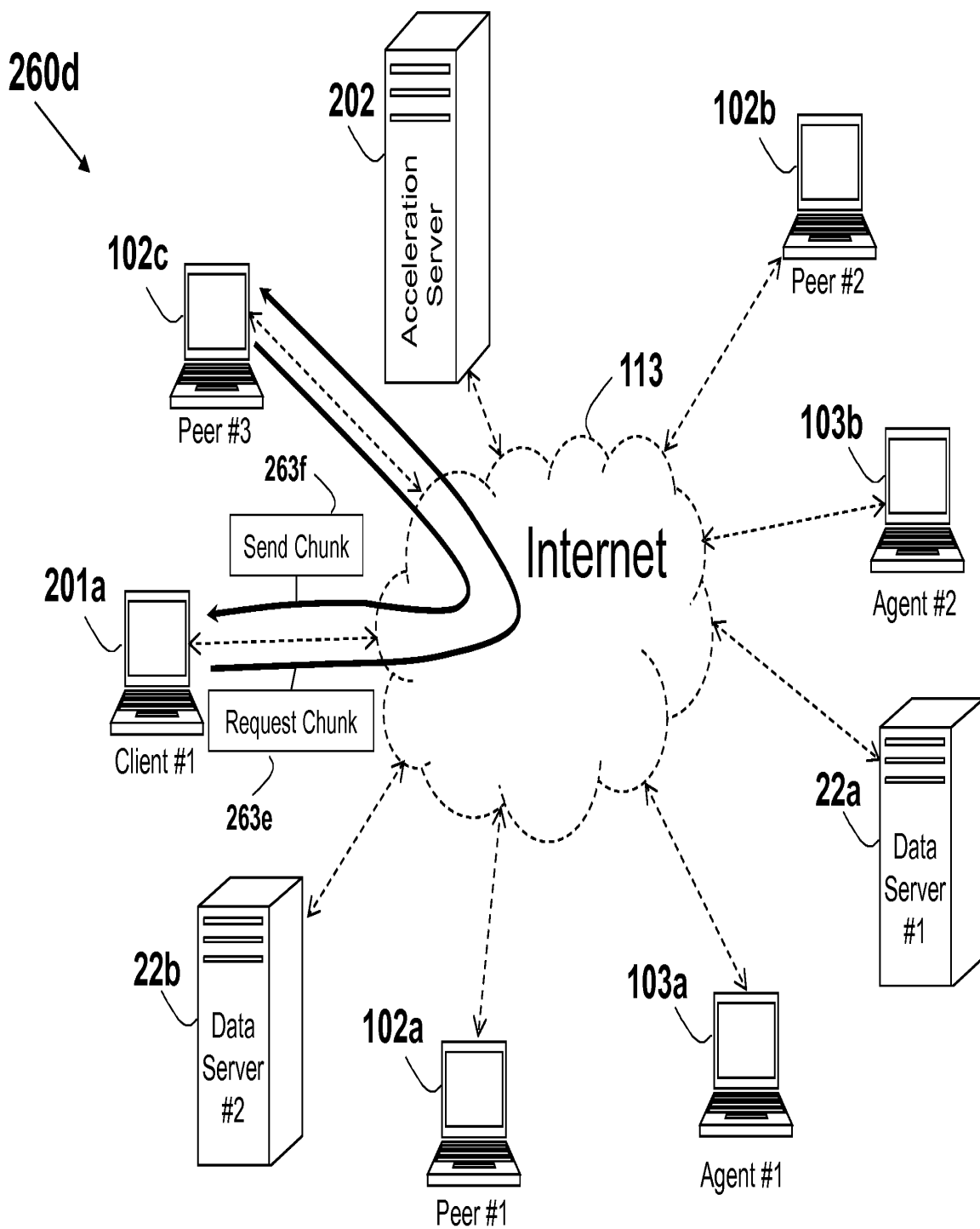


FIG. 26d

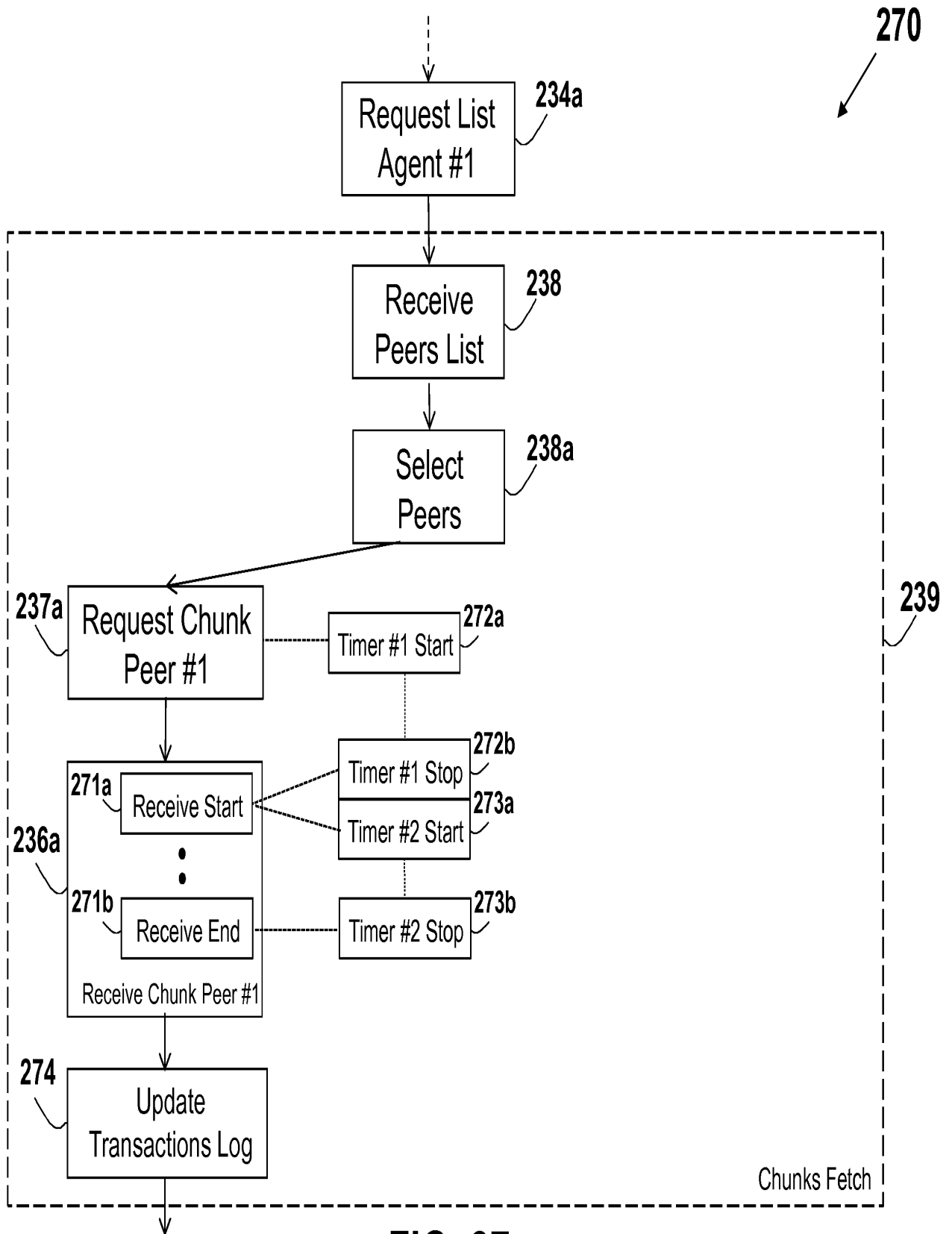


FIG. 27

280



281a



281b



281c



281d



281e



281f



282 →

282a →

282b →

282c →

282d →

DATE / TIME	PEER IP ADDRESS	SERVER IP ADDRESS	URL	BW (Kb/s)	RTT (ms)
13/3 9:23	229.155.81.168	128.164.35.142	www.111.com/22.mpg	1000	30
13/3 9:46	248.107.109.10	49.154.2.5	www.xxx.com/hy.avi	350	70
15/4 11:22	158.217.19.195	72.251.238.51	www.yyy.com/t6.php	2500	540
16/4 16:43	253.6.96.135	169.101.27.217	www.1er.com/5.shtml	1400	170

FIG. 28

280a



283a



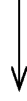
283b



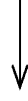
283c



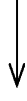
283d



283e



283f



284 →

284a →

284b →

284c →

284d →

DATE / TIME	TUNNEL IP ADDRESS	SERVER IP ADDRESS	URL	BW (Kb/s)	RTT (ms)
13/3 9:23	139.230.154.213	49.154.2.5	https://www.youtube.com/ watch?v=cJwNwCD6CAE	1000	30
13/3 9:46	132.171.60.197	49.154.2.5	https://www.youtube.com/ watch?v=FRAbW6oCgHM	350	70
15/4 11:22	248.46.80.36	49.154.2.5	https://www.youtube.com/ watch?v=CQQIEQh8pU0	2500	540
16/4 16:43	31.16.208.171	49.154.2.5	https://www.youtube.com/ watch?v=KJbtX1xWyMA	1400	170

FIG. 28a

290

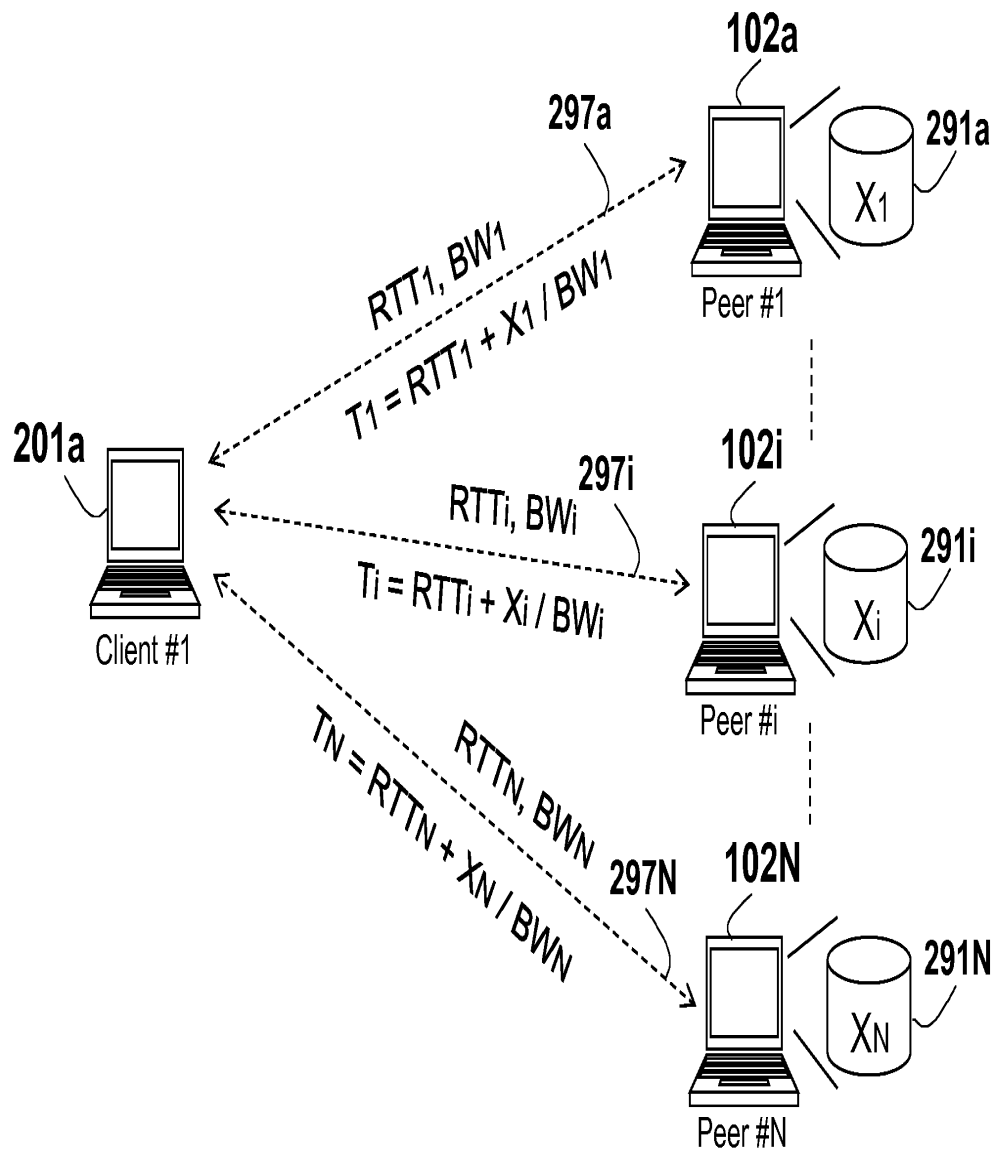


FIG. 29

$$(1) \text{ Total Latency: } T = \max_i (T_i) = \max_i (RTT_i + X_i / BW_i) \quad \leftarrow 292a$$

$$(2) \text{ Target: Minimize } (T) \text{ subject to : } X = \sum_{i=1}^N X_i \quad \leftarrow 292b$$

$$(3) T = T_1 = T_2 = \dots = T_N \quad \leftarrow 292c$$

$$(4) X_i = BW_i * [((X + \sum_{i=1}^N RTT_i * BW_i) / \sum_{i=1}^N BW_i) - RTT_i] \quad \leftarrow 292d$$

$$(5) T = T_1 = \dots = T_N = [X + \sum_{i=1}^N (RTT_i * BW_i)] / \sum_{i=1}^N BW_i \quad \leftarrow 292e$$

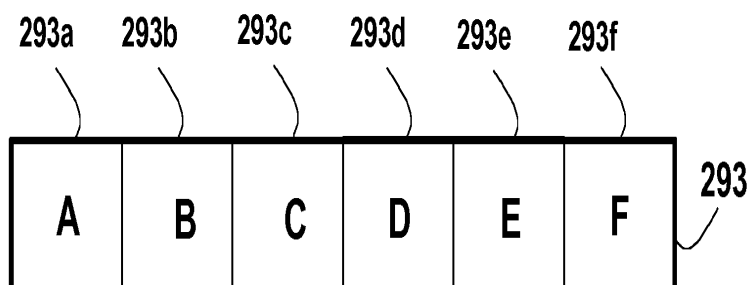
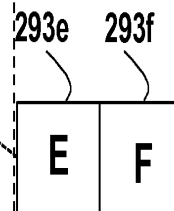
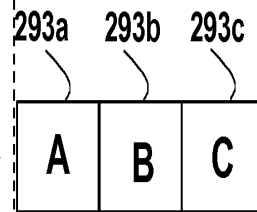


FIG. 29a

295a



294

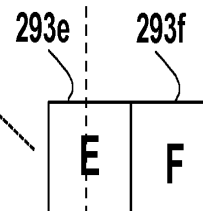
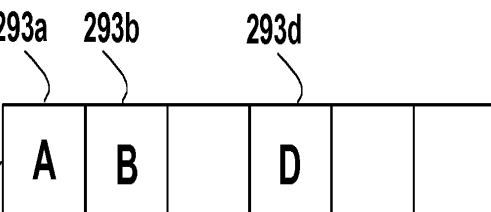


t = 0

295b



294



t = t1

FIG. 29b

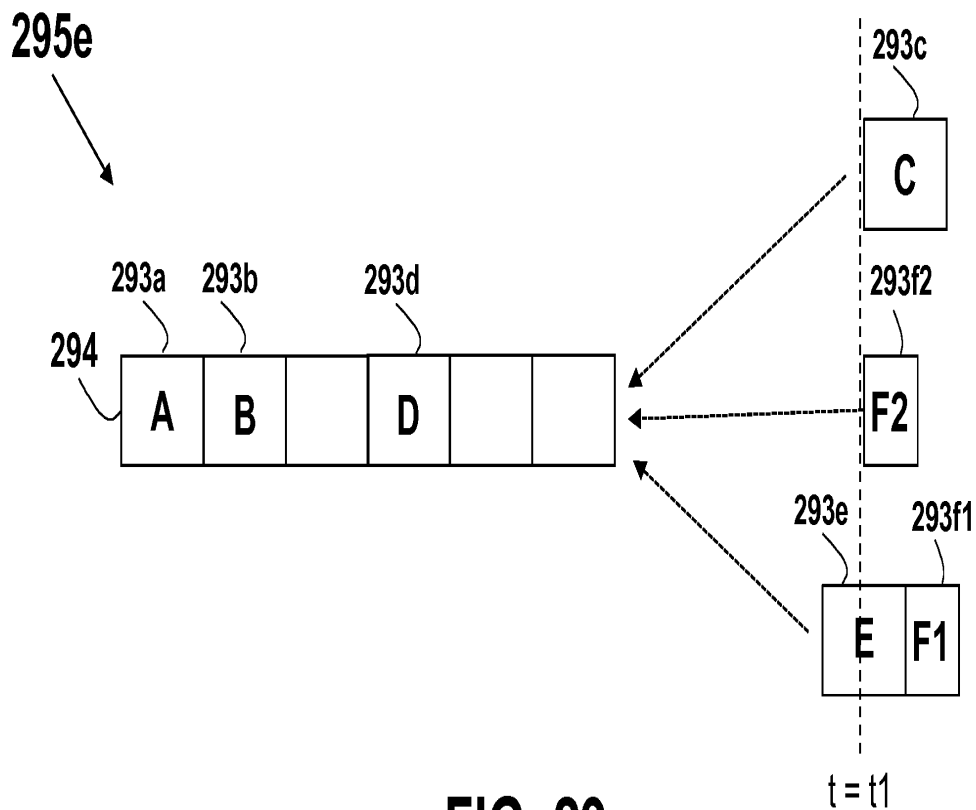
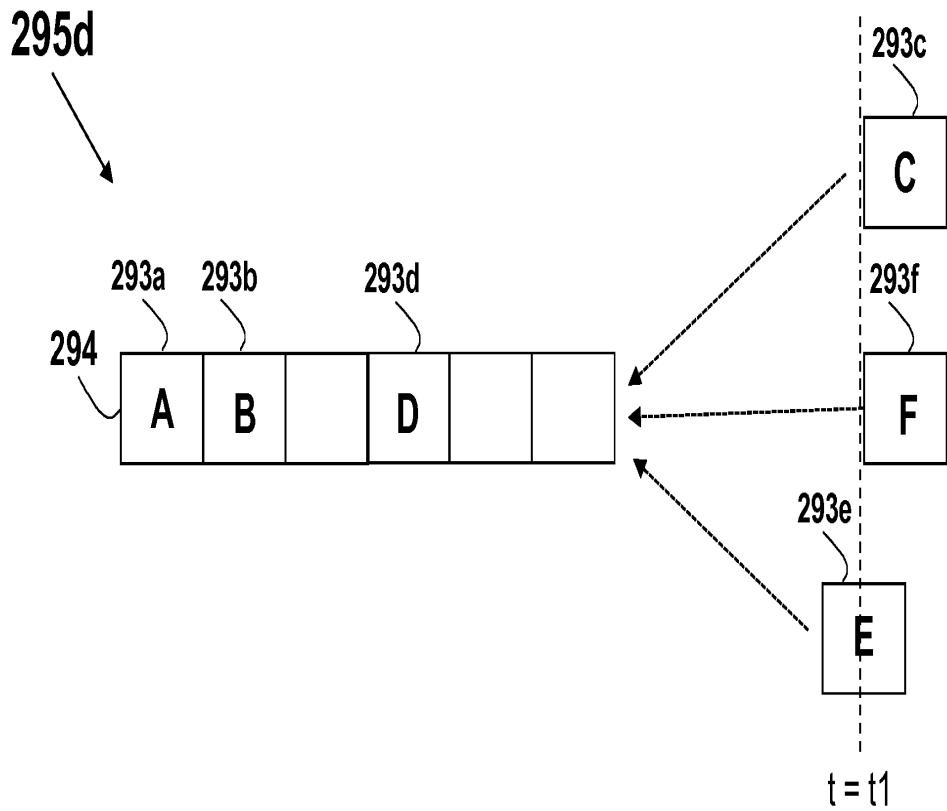


FIG. 29c

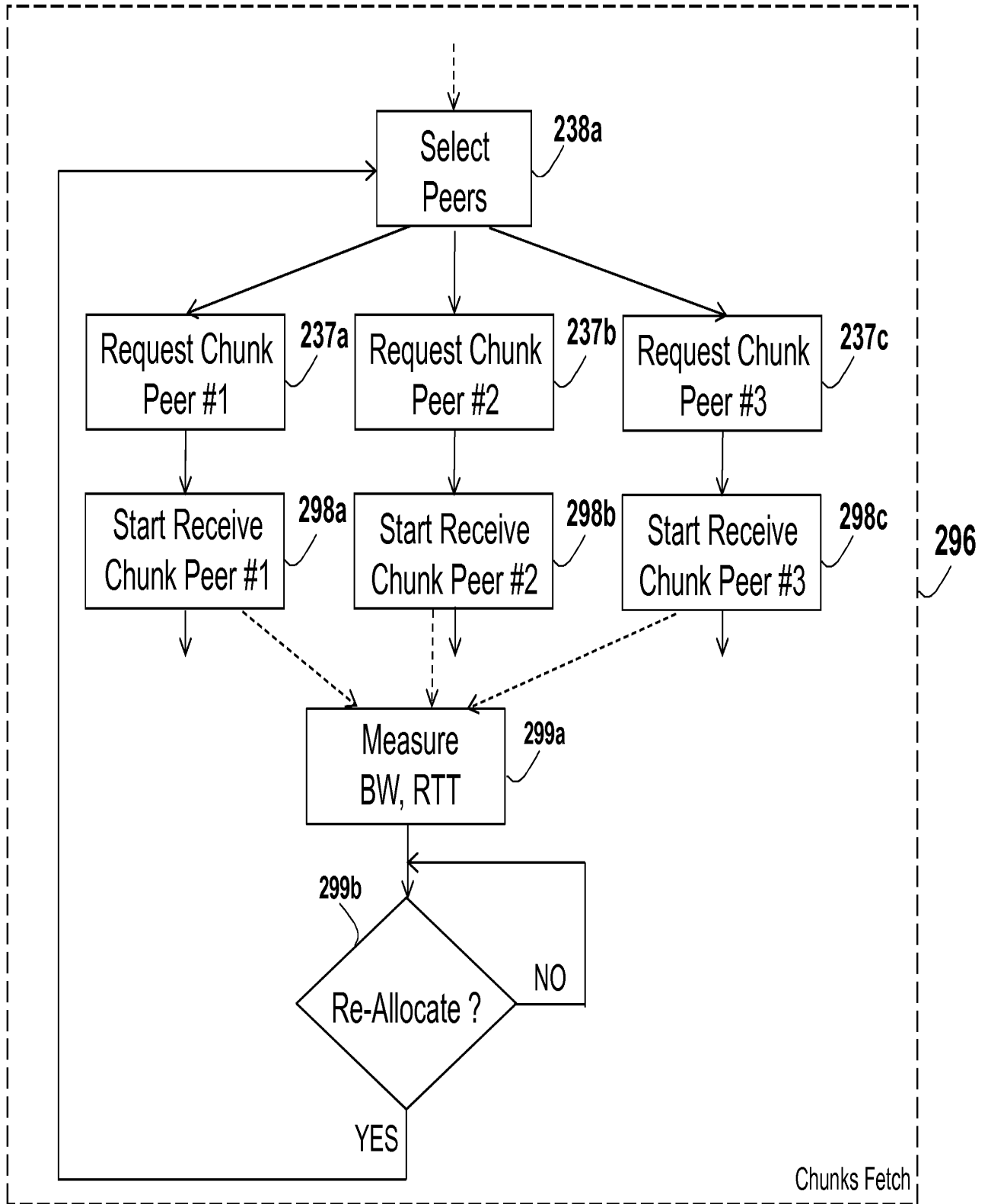


FIG. 29d

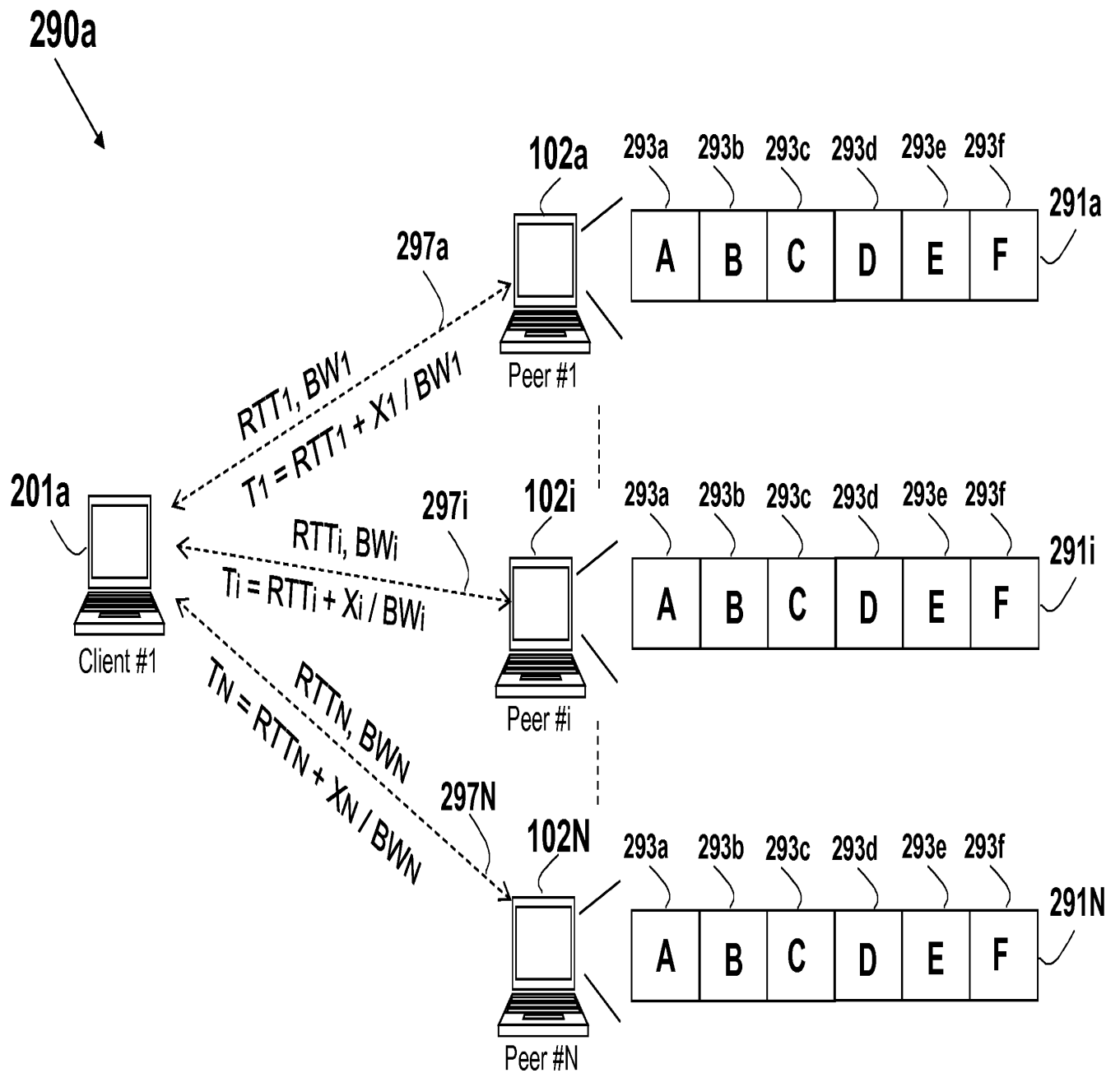


FIG. 29e

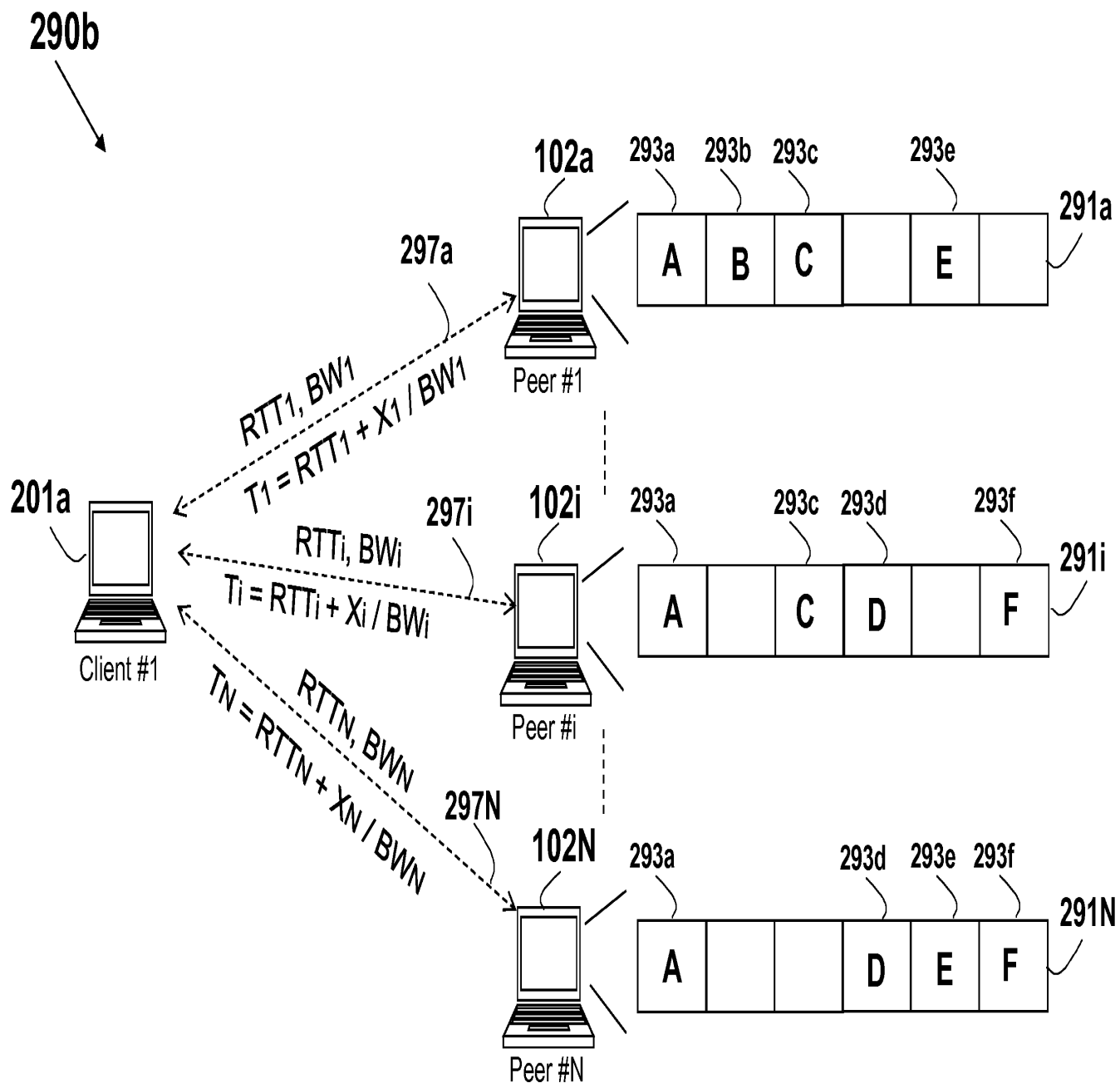


FIG. 29f

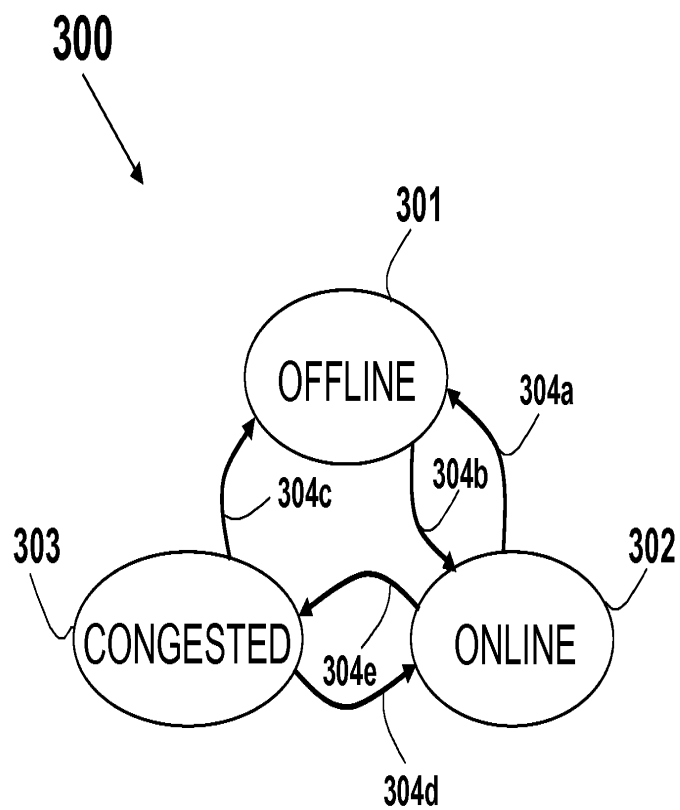


FIG. 30

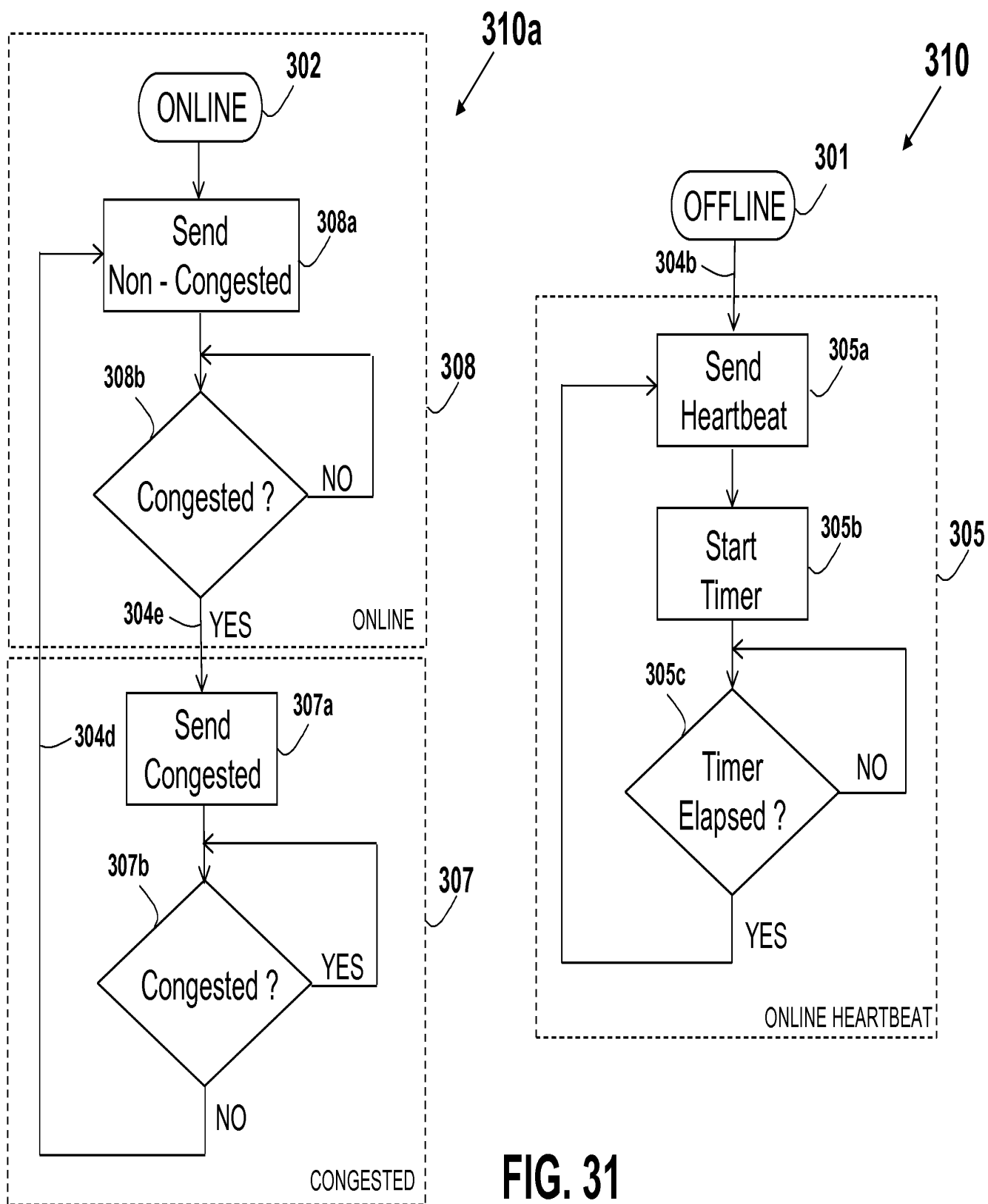


FIG. 31

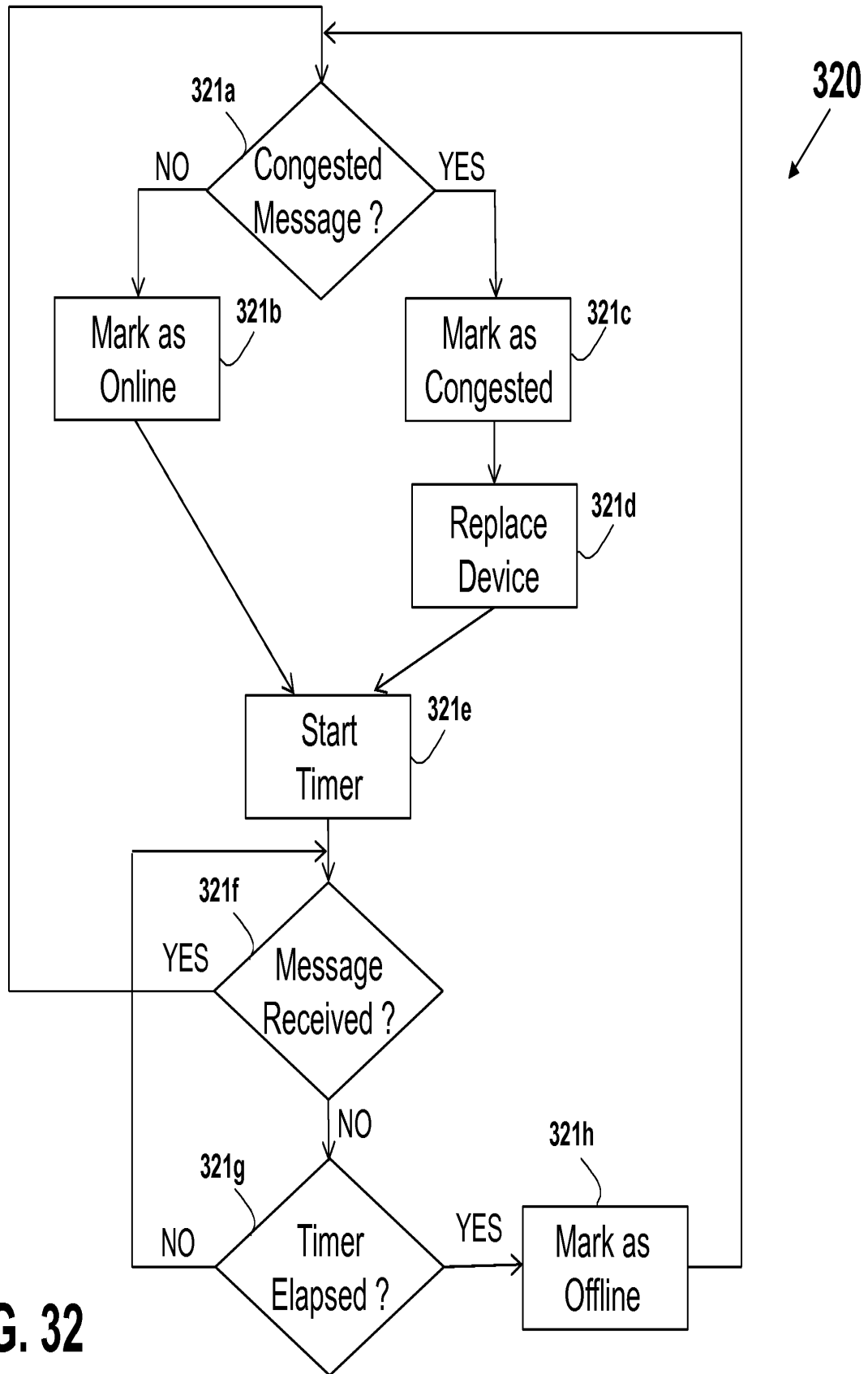


FIG. 32

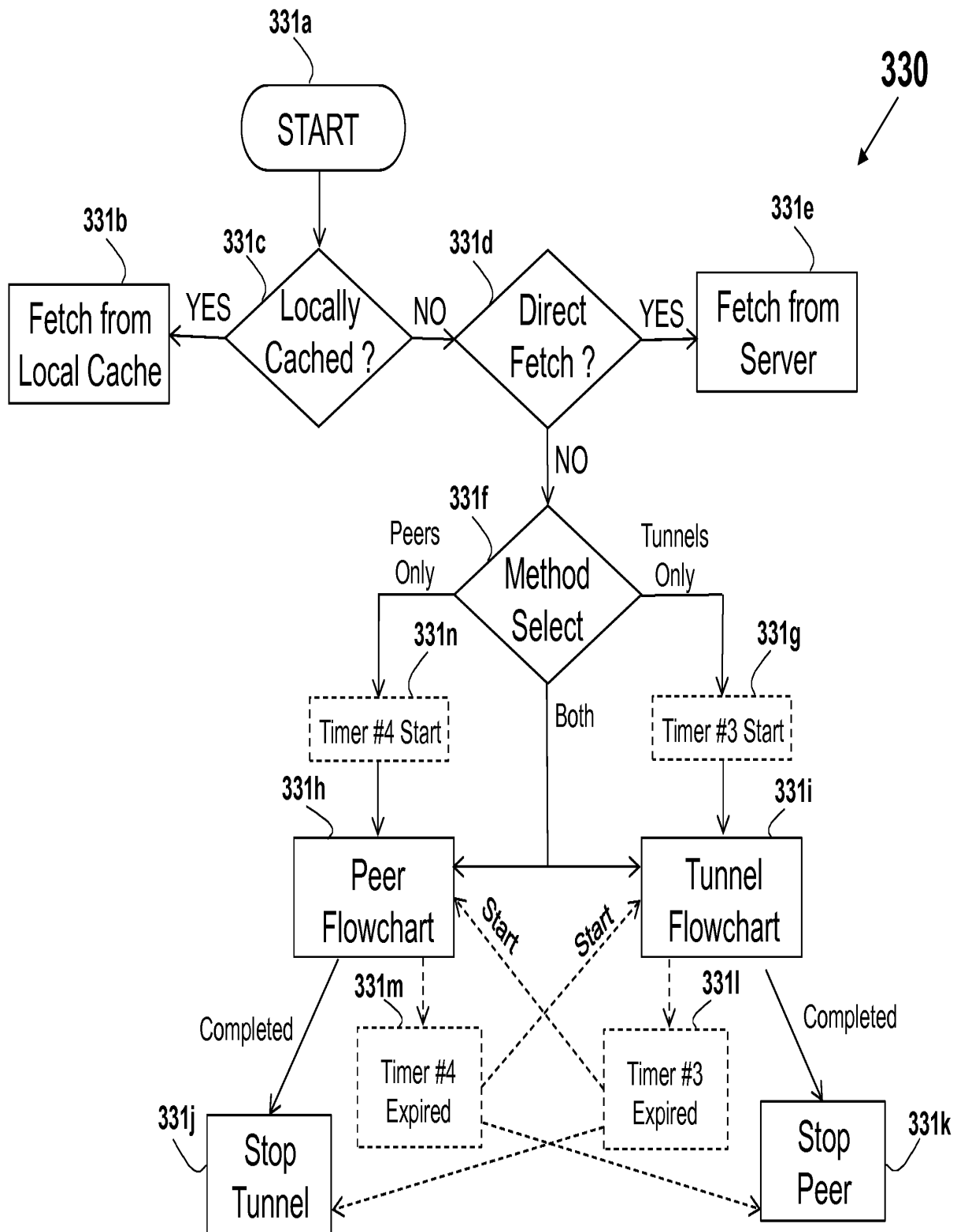


FIG. 33

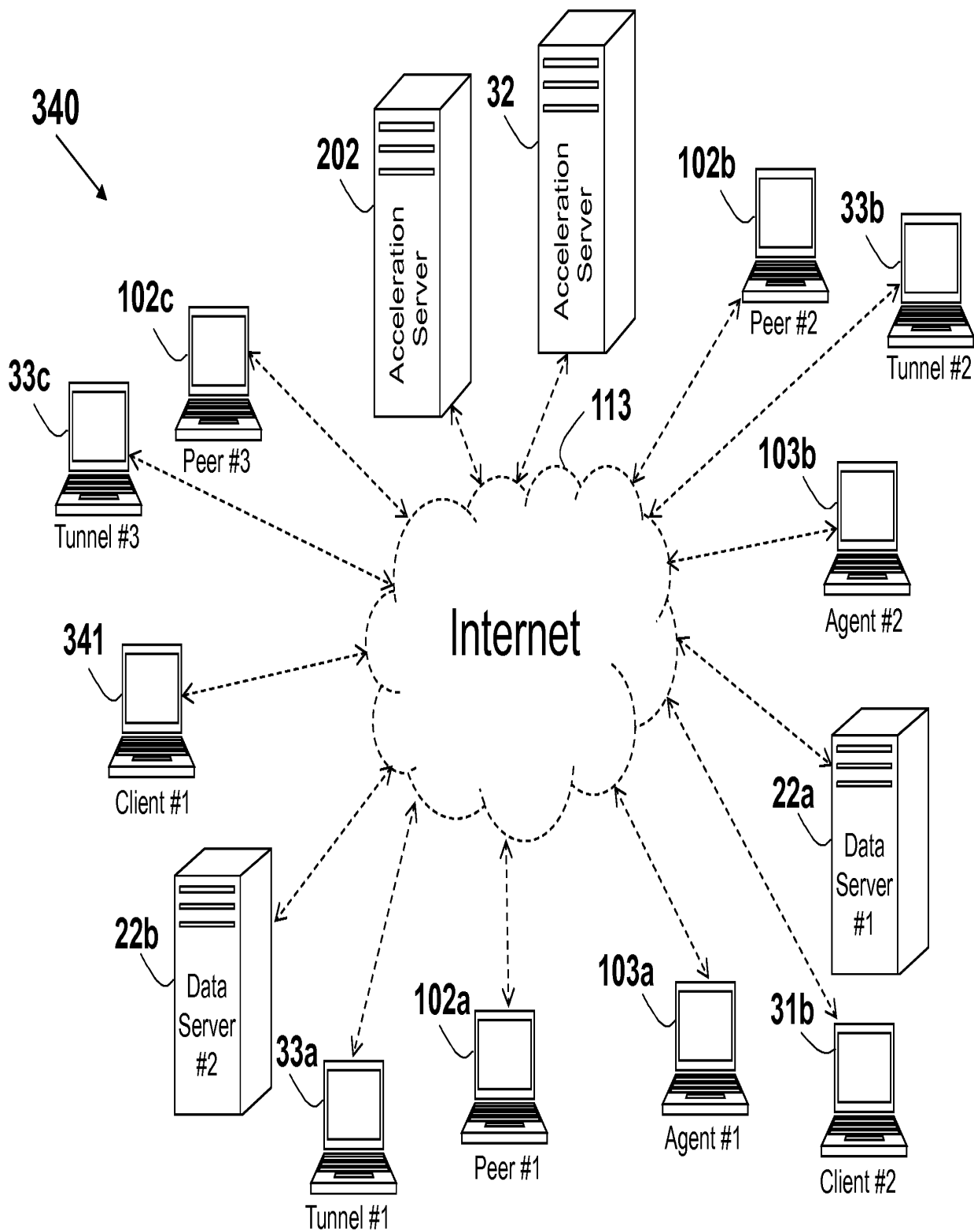


FIG. 34

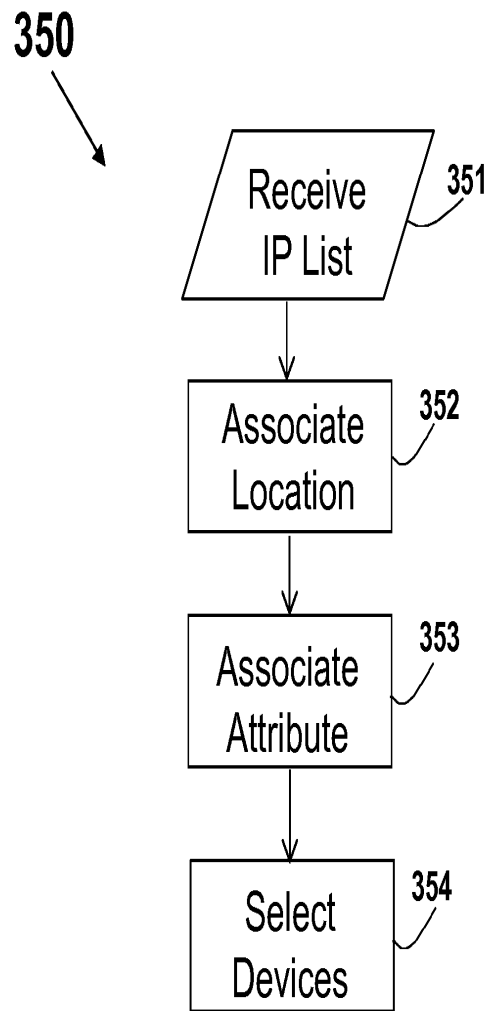


FIG. 35

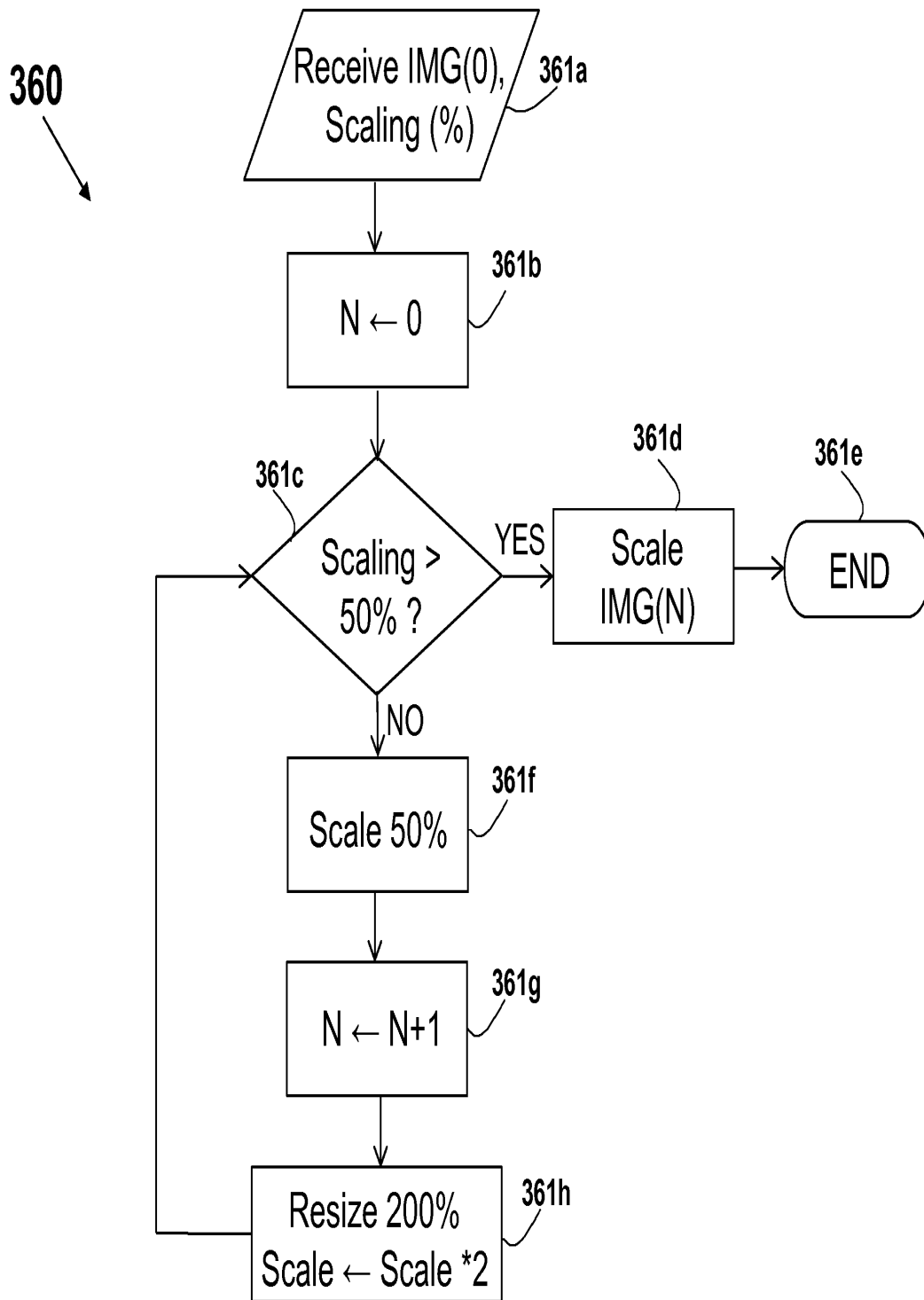


FIG. 36

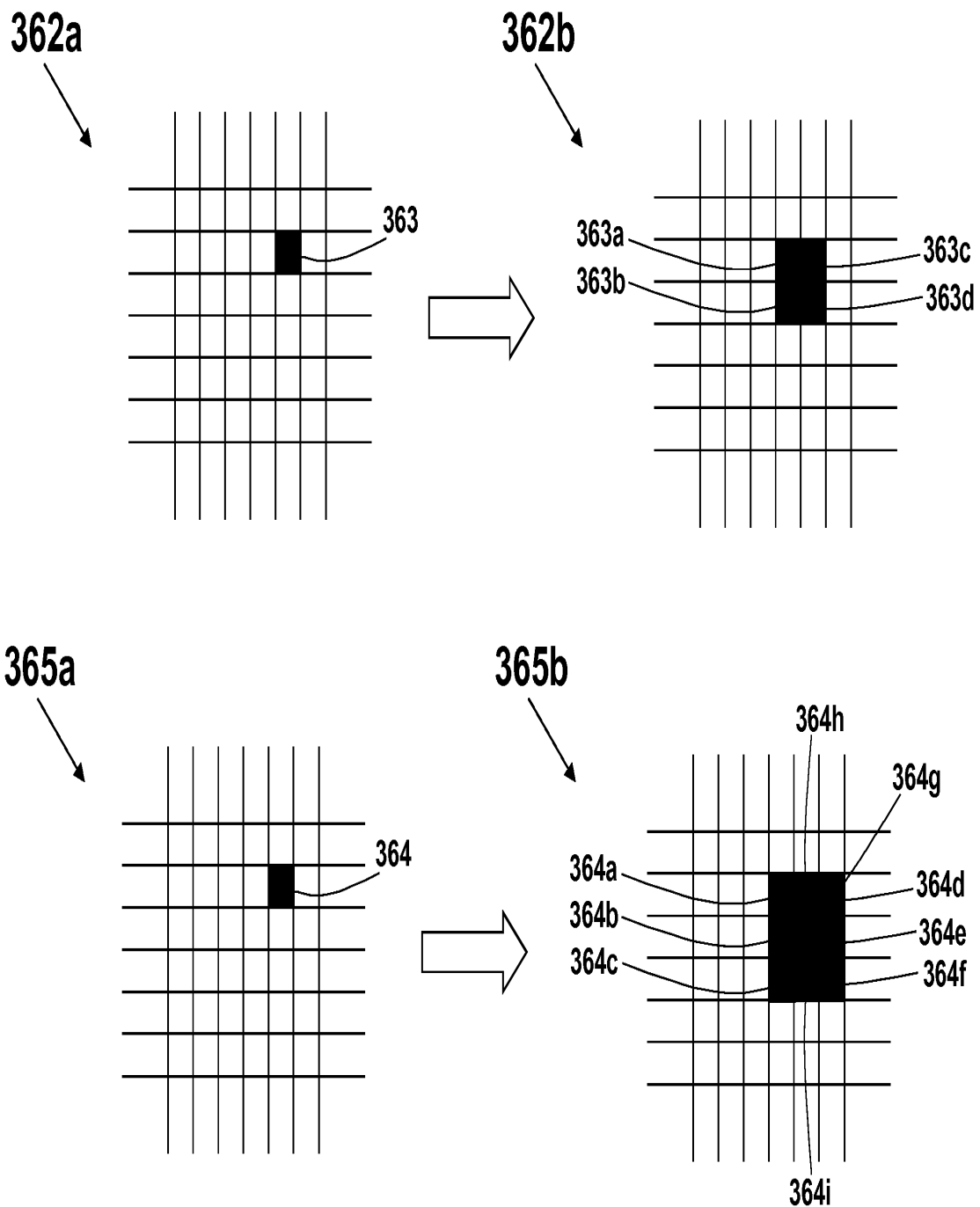


FIG. 36a

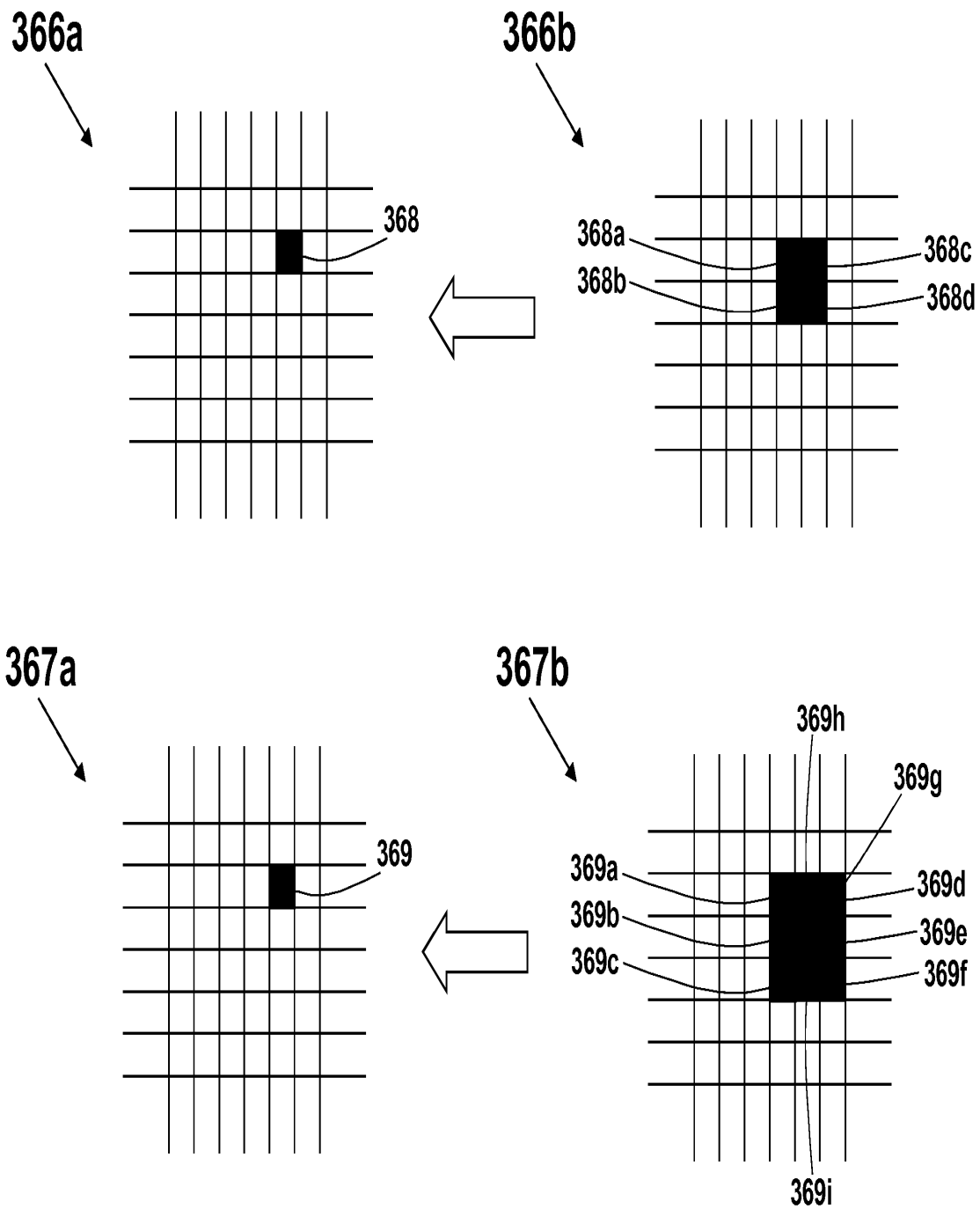
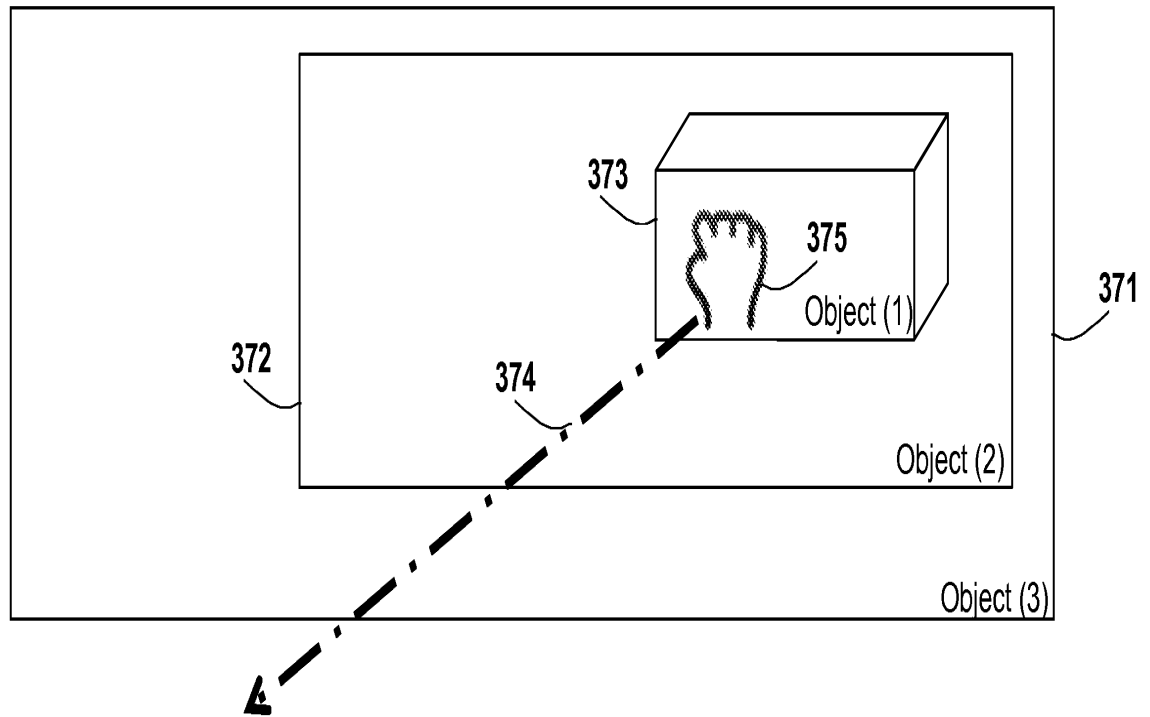


FIG. 36b

370a



370b

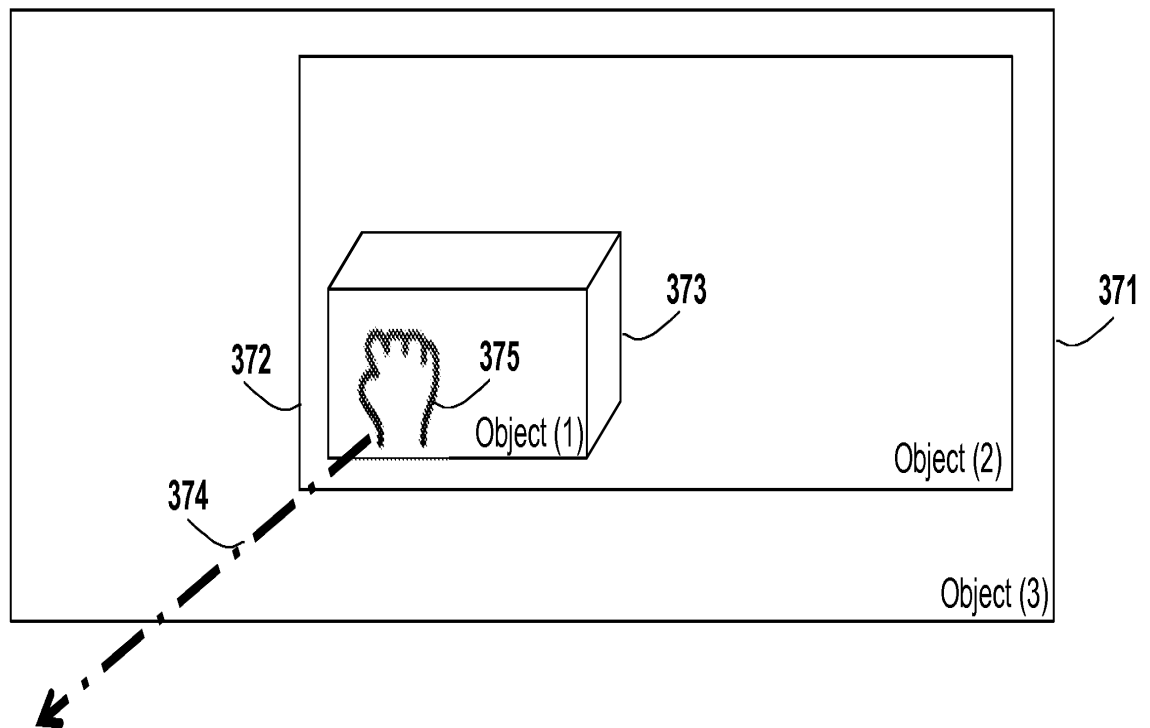


FIG. 37

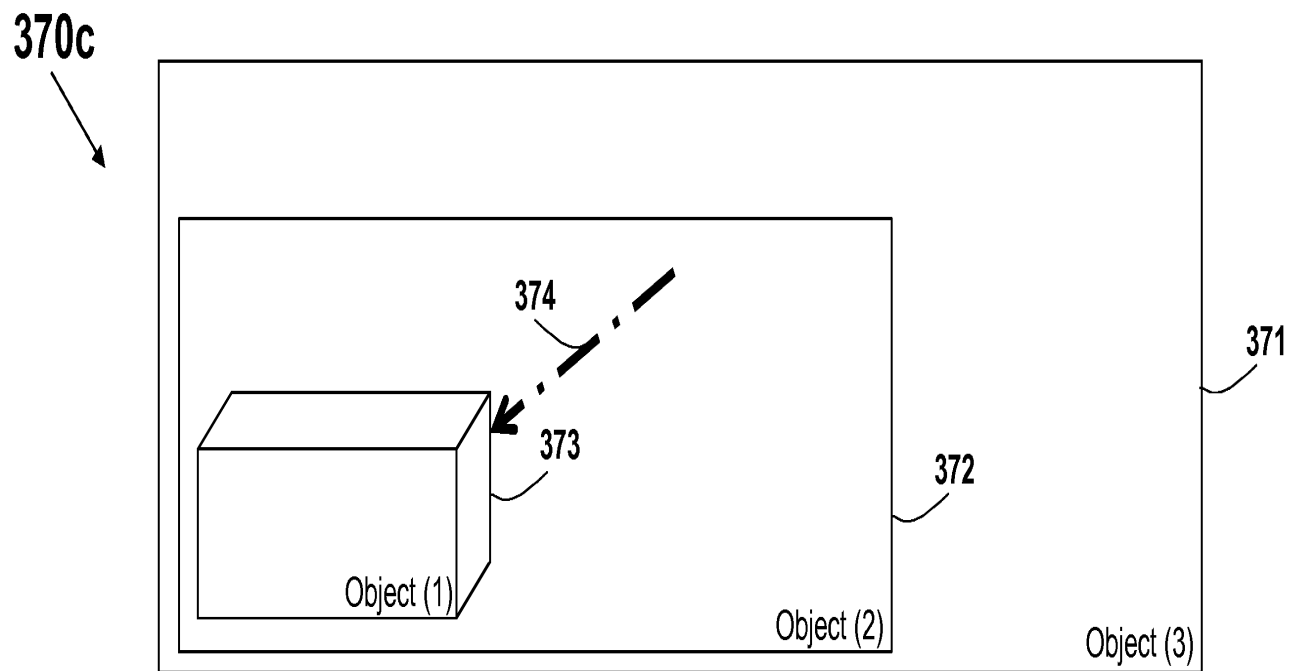


FIG. 37a

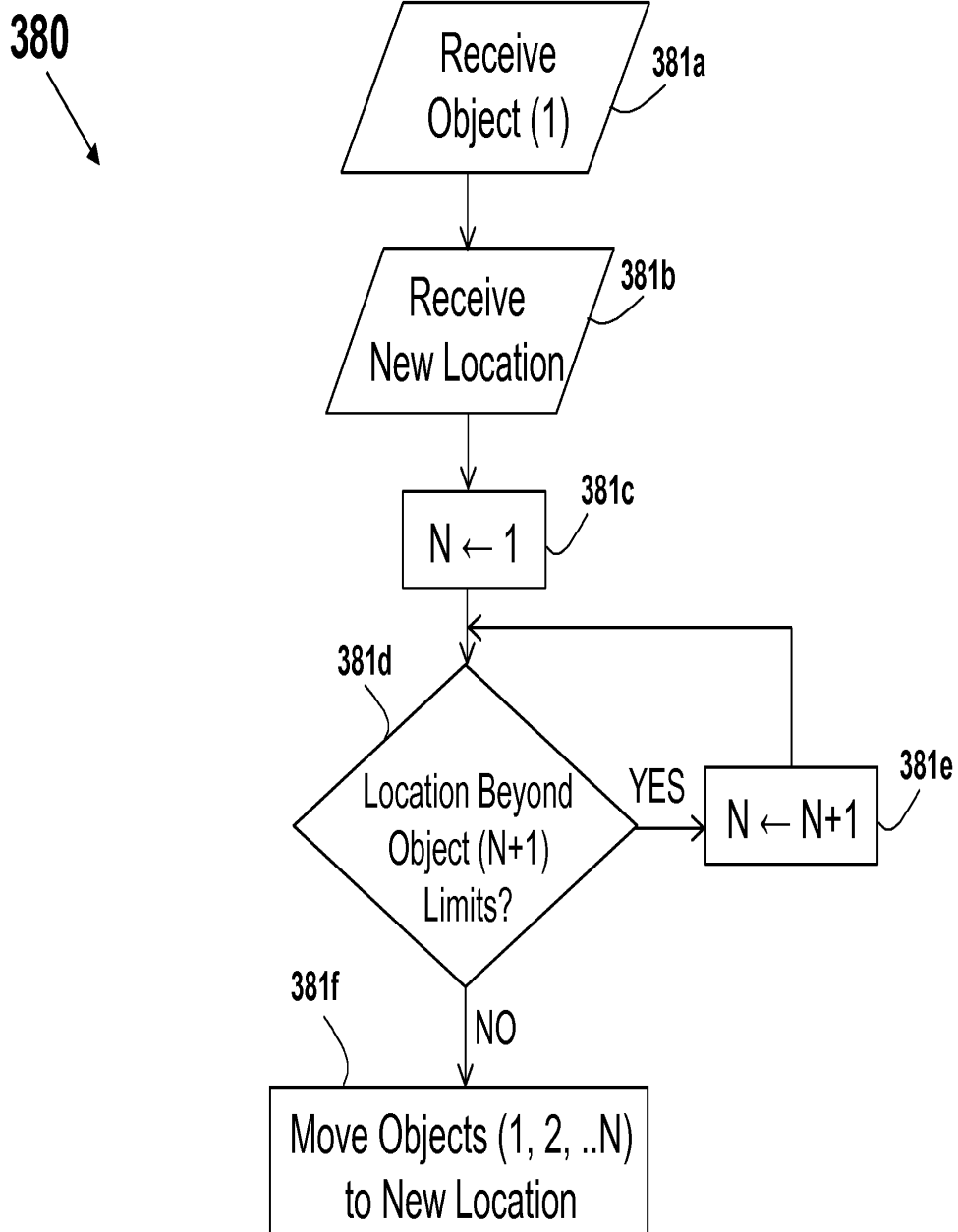


FIG. 38

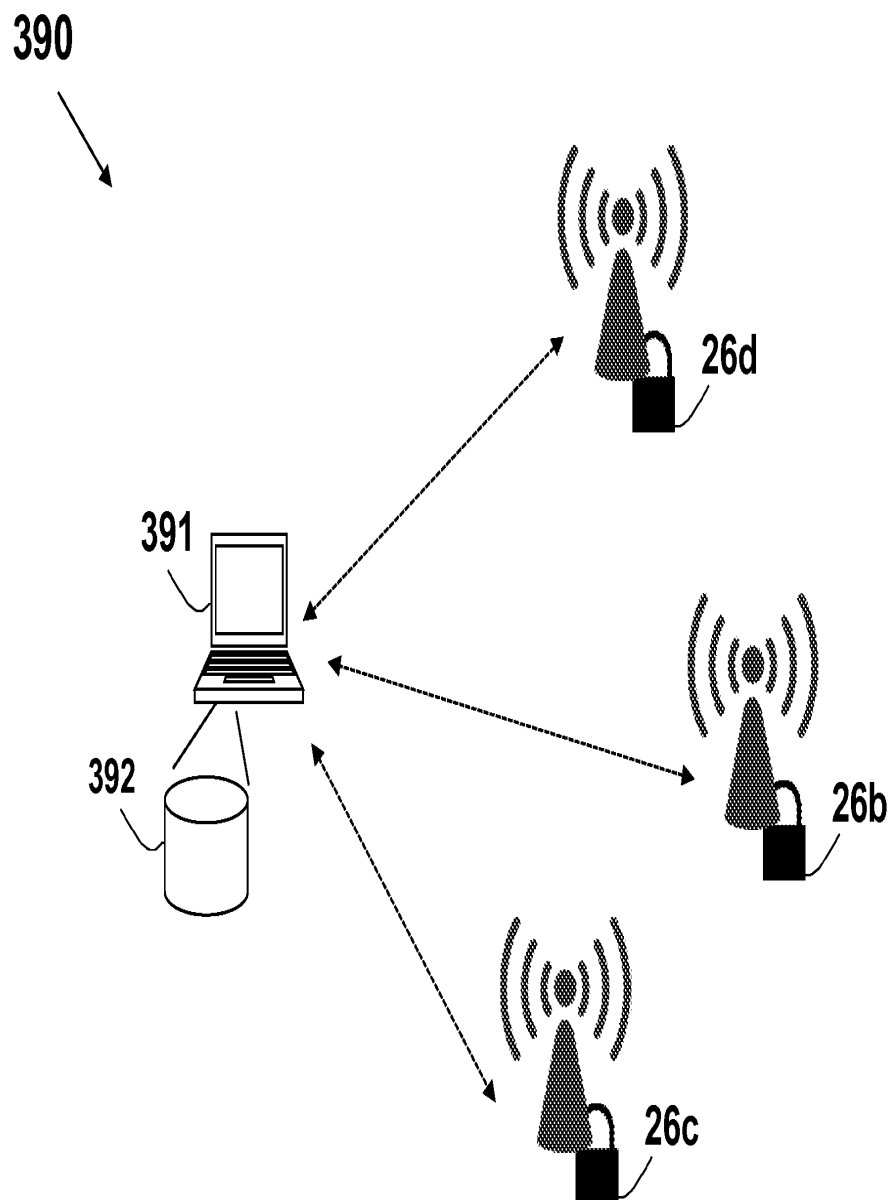


FIG. 39

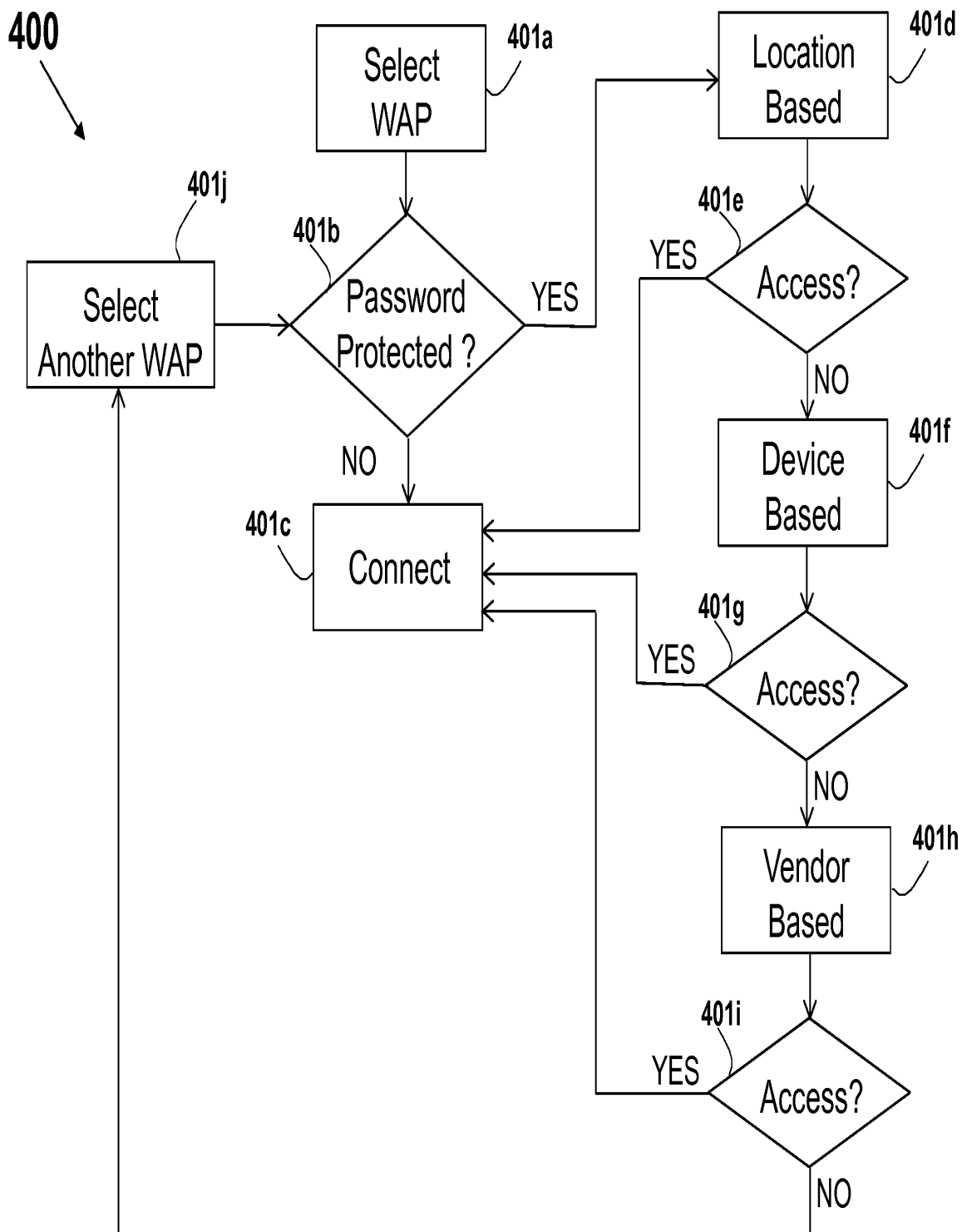


FIG. 40

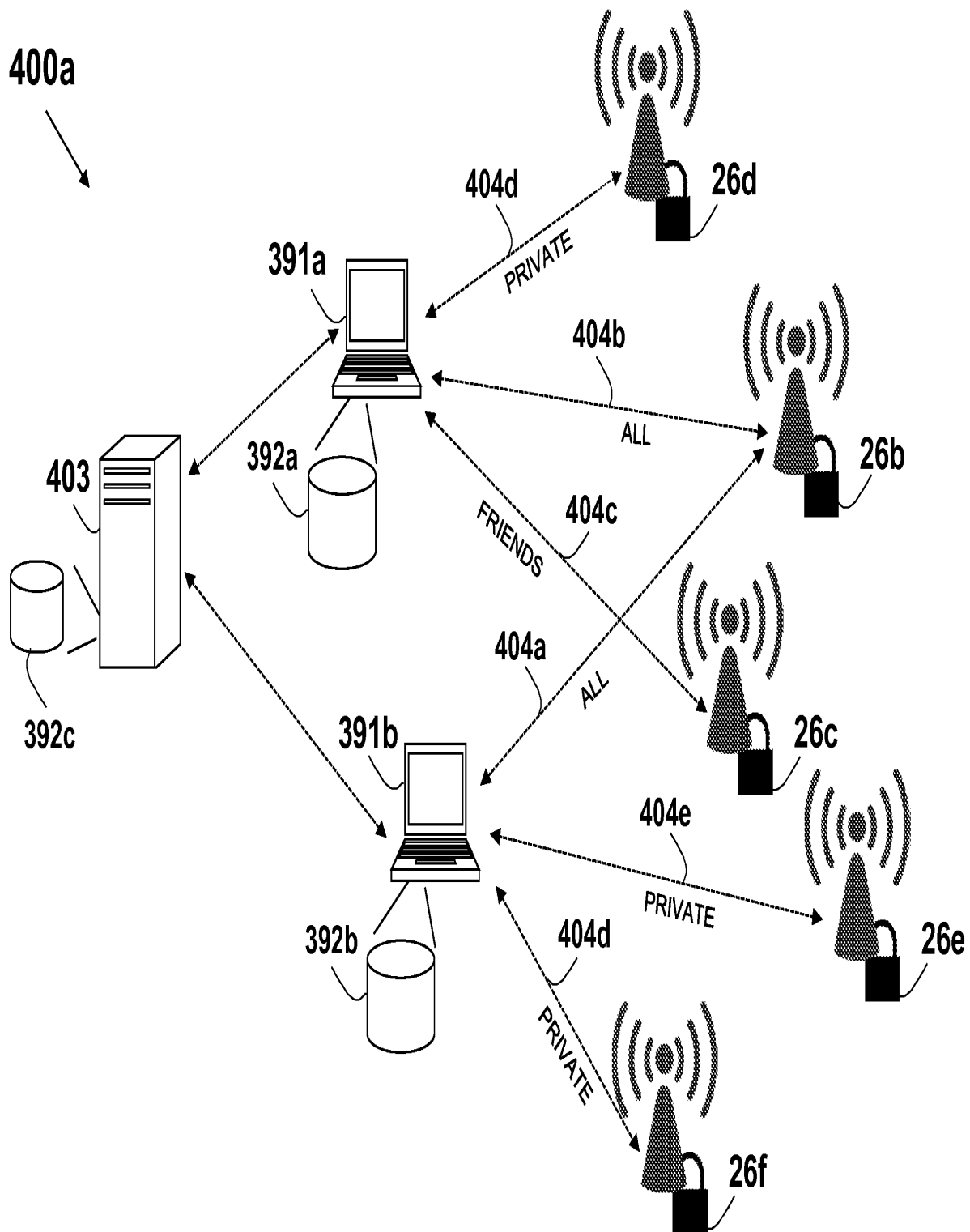


FIG. 41

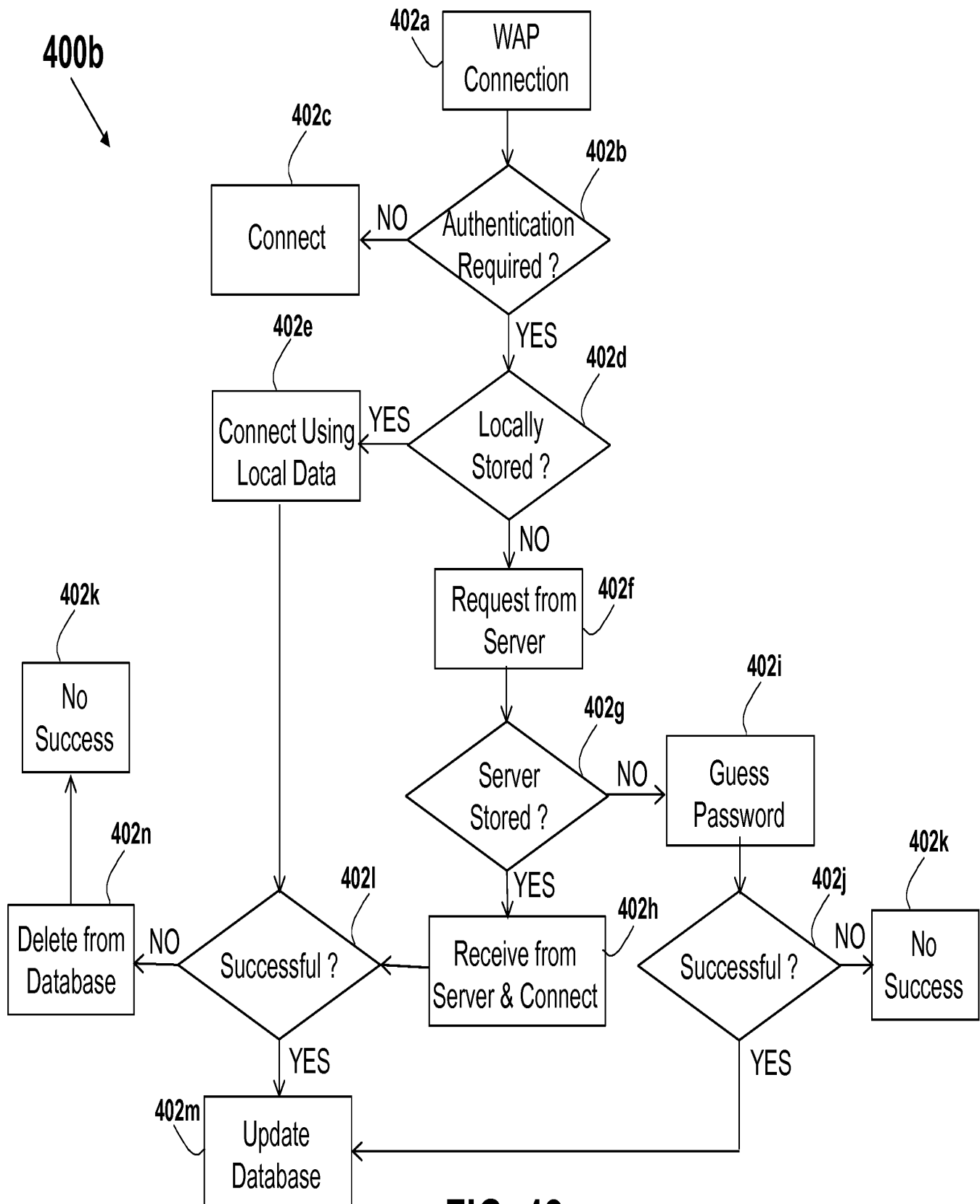


FIG. 42

410

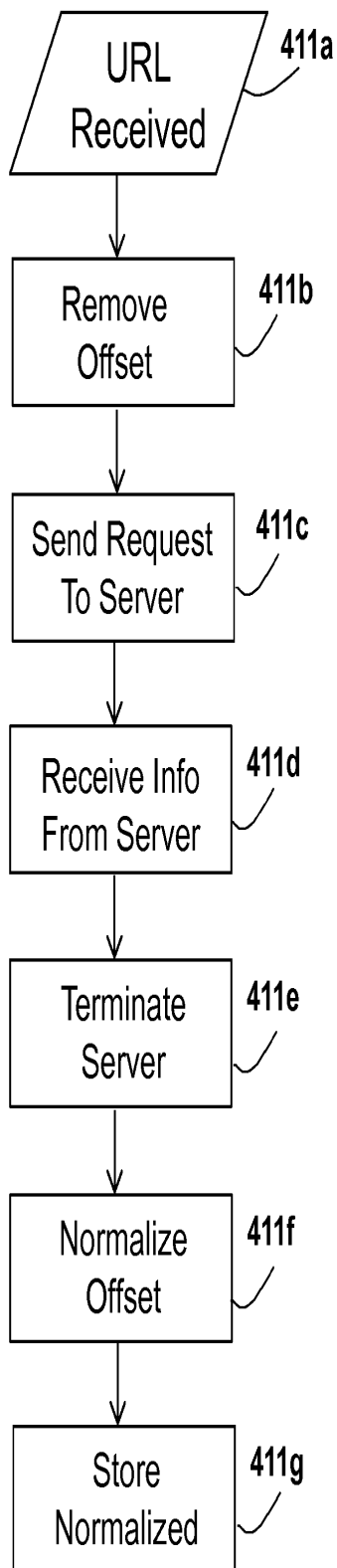


FIG. 43

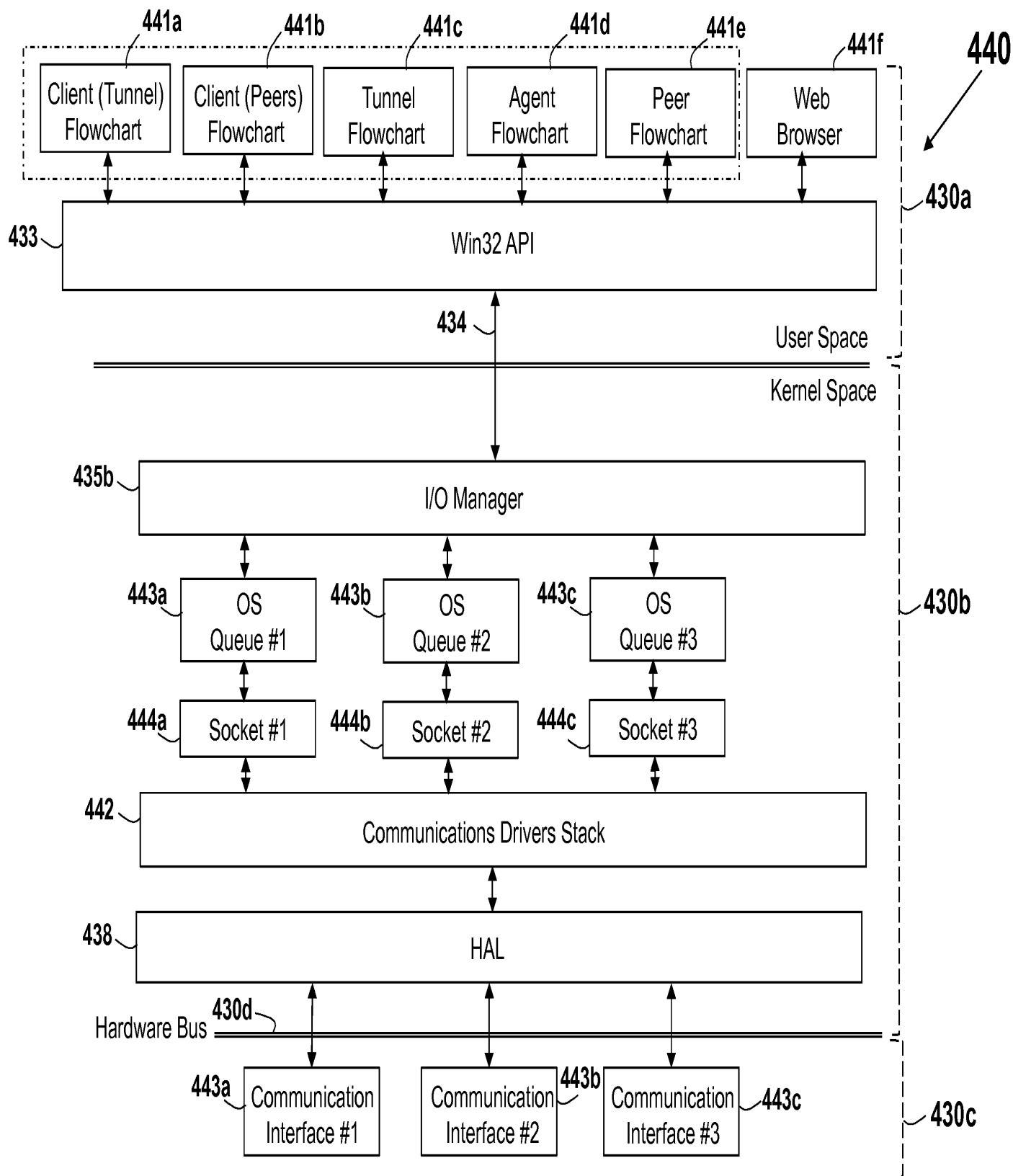


FIG. 44

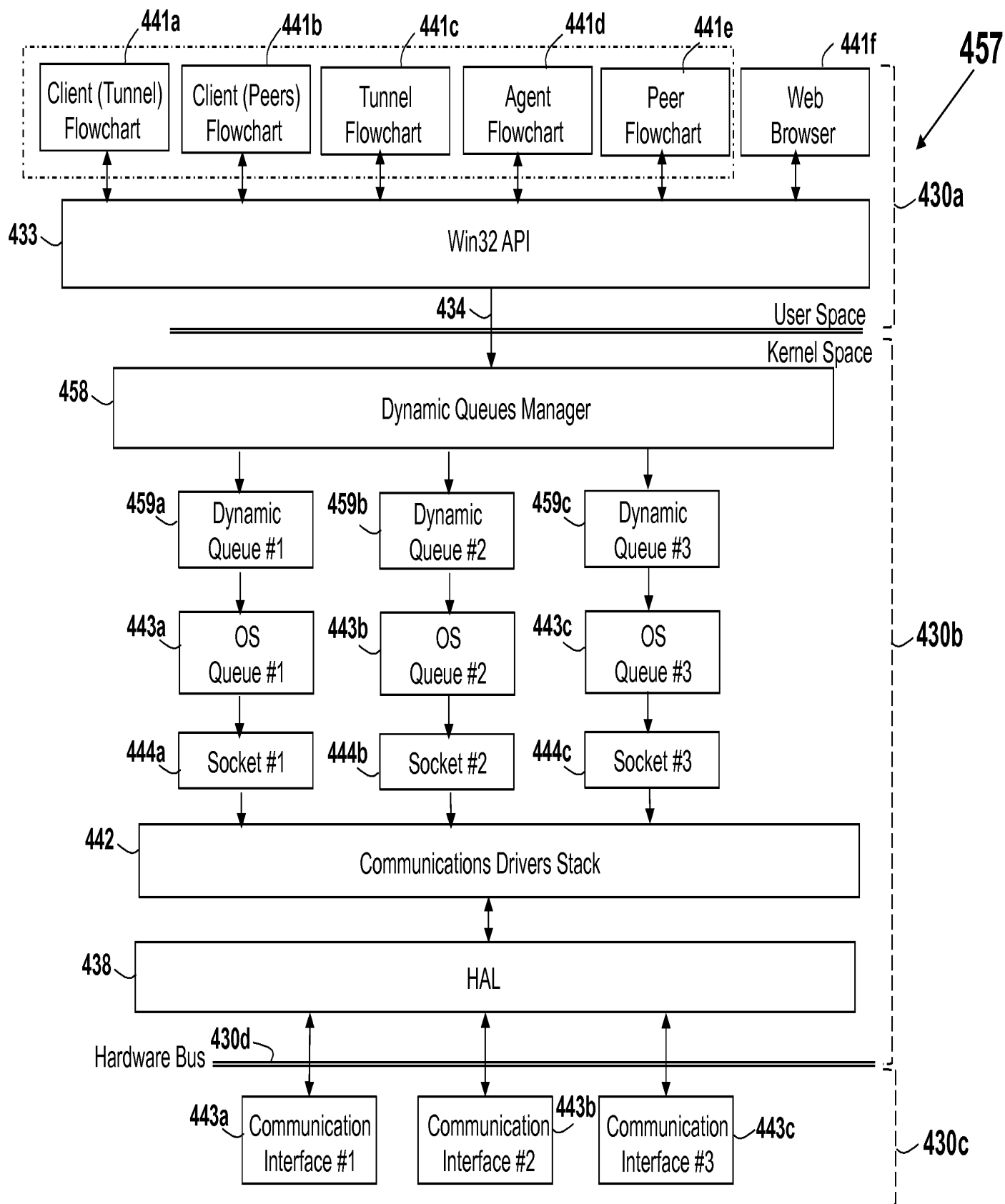


FIG. 45

460

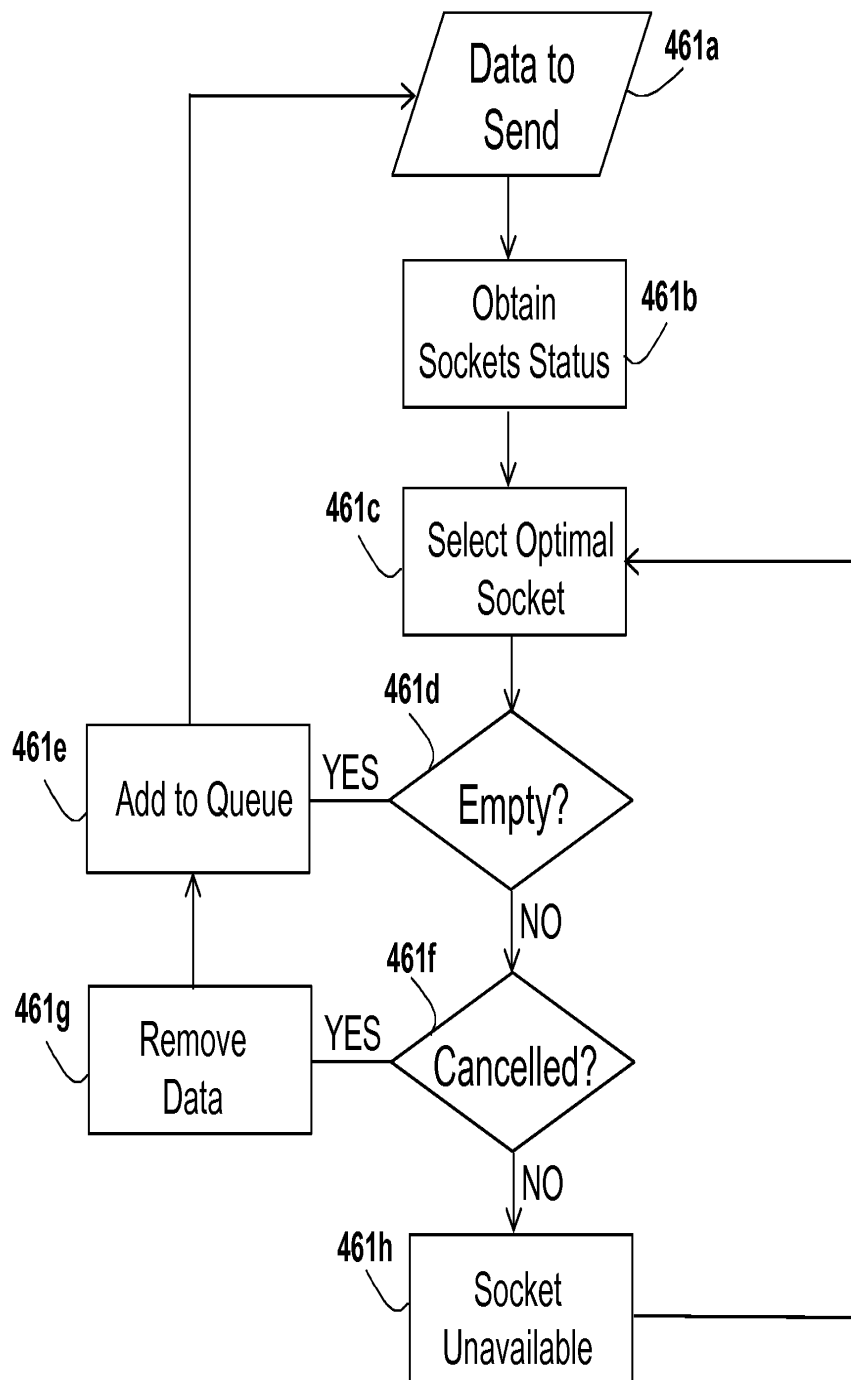


FIG. 46

470b

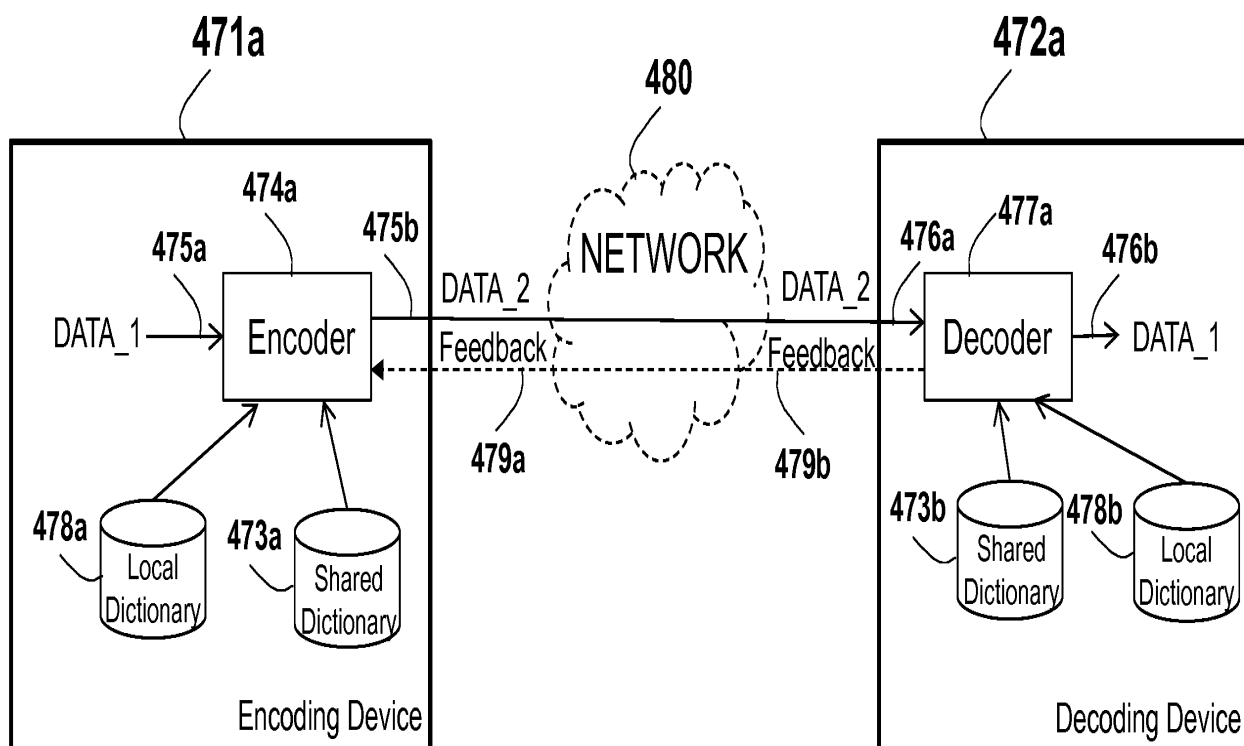


FIG. 47

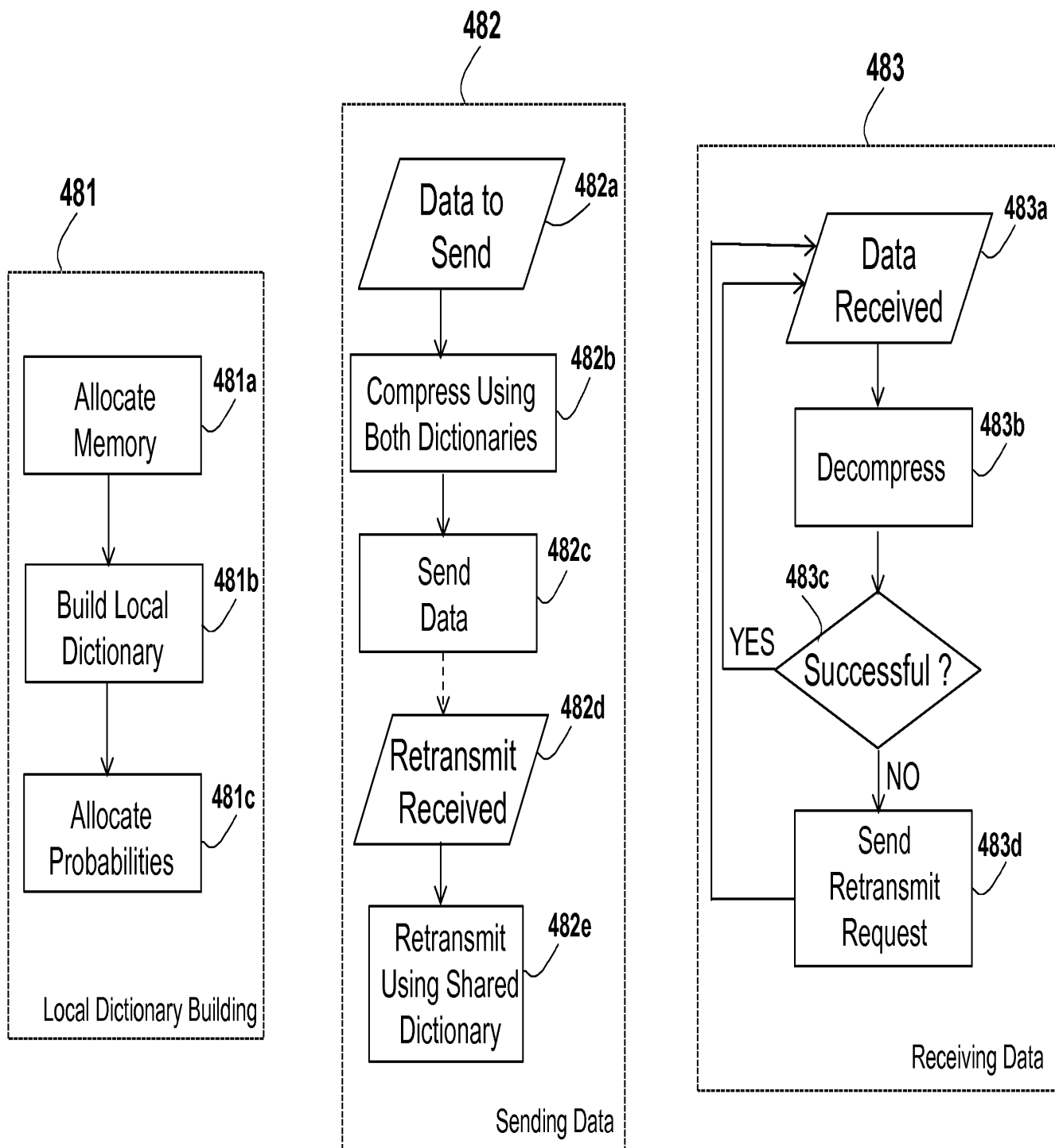


FIG. 48

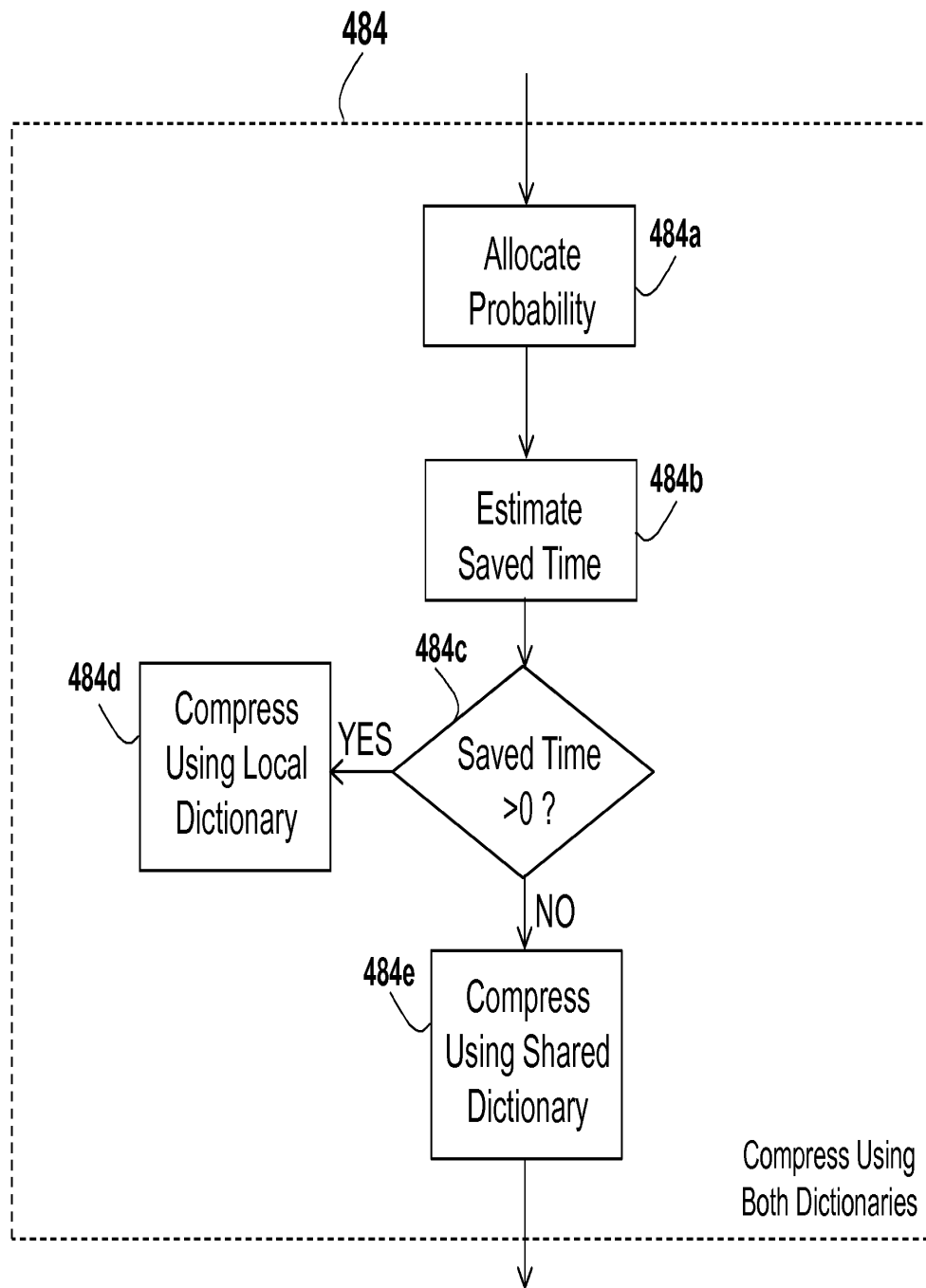


FIG. 48a

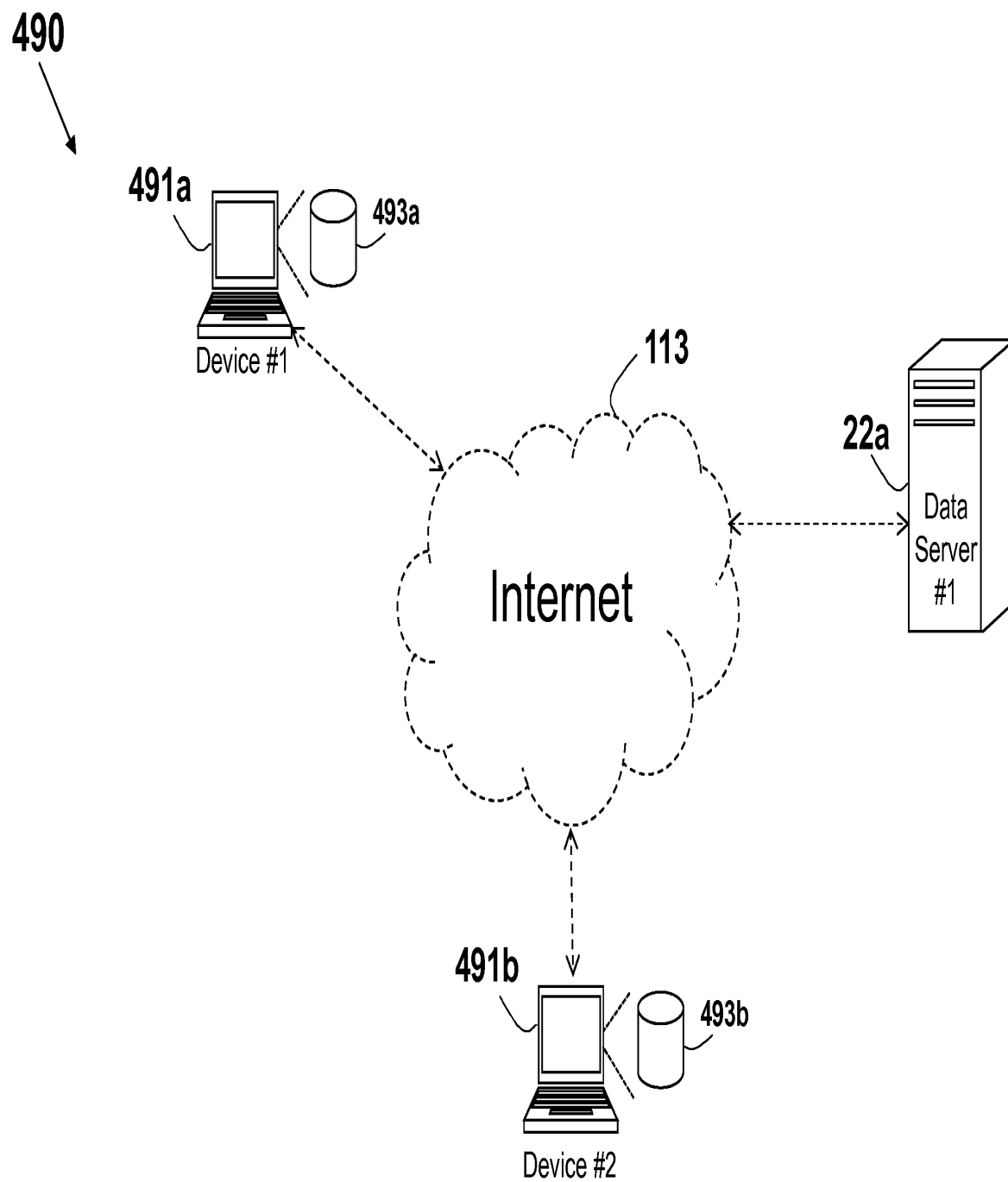


FIG. 49

490a

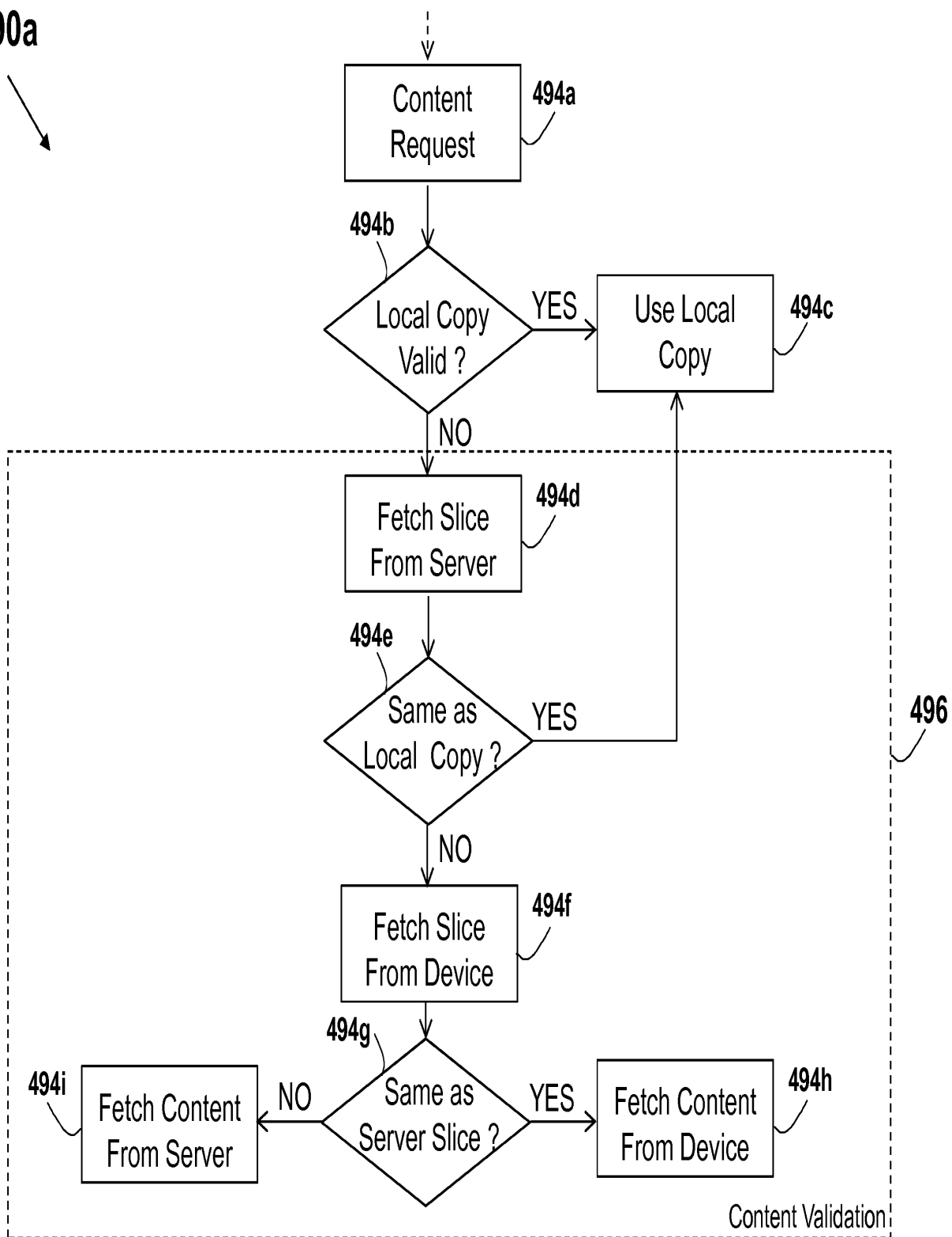


FIG. 49a

490b

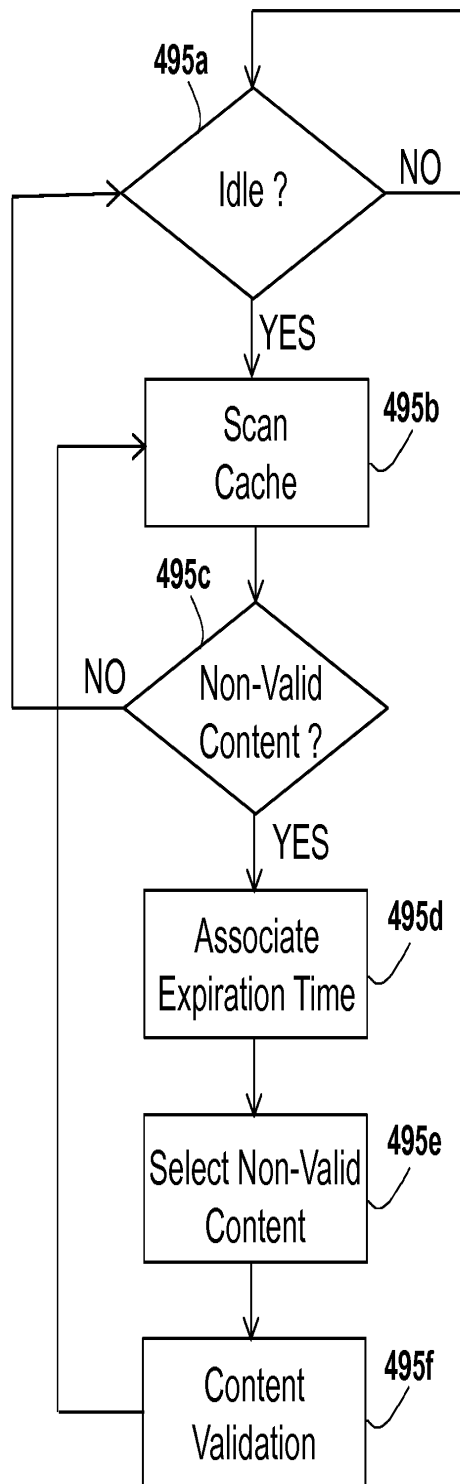


FIG. 49b

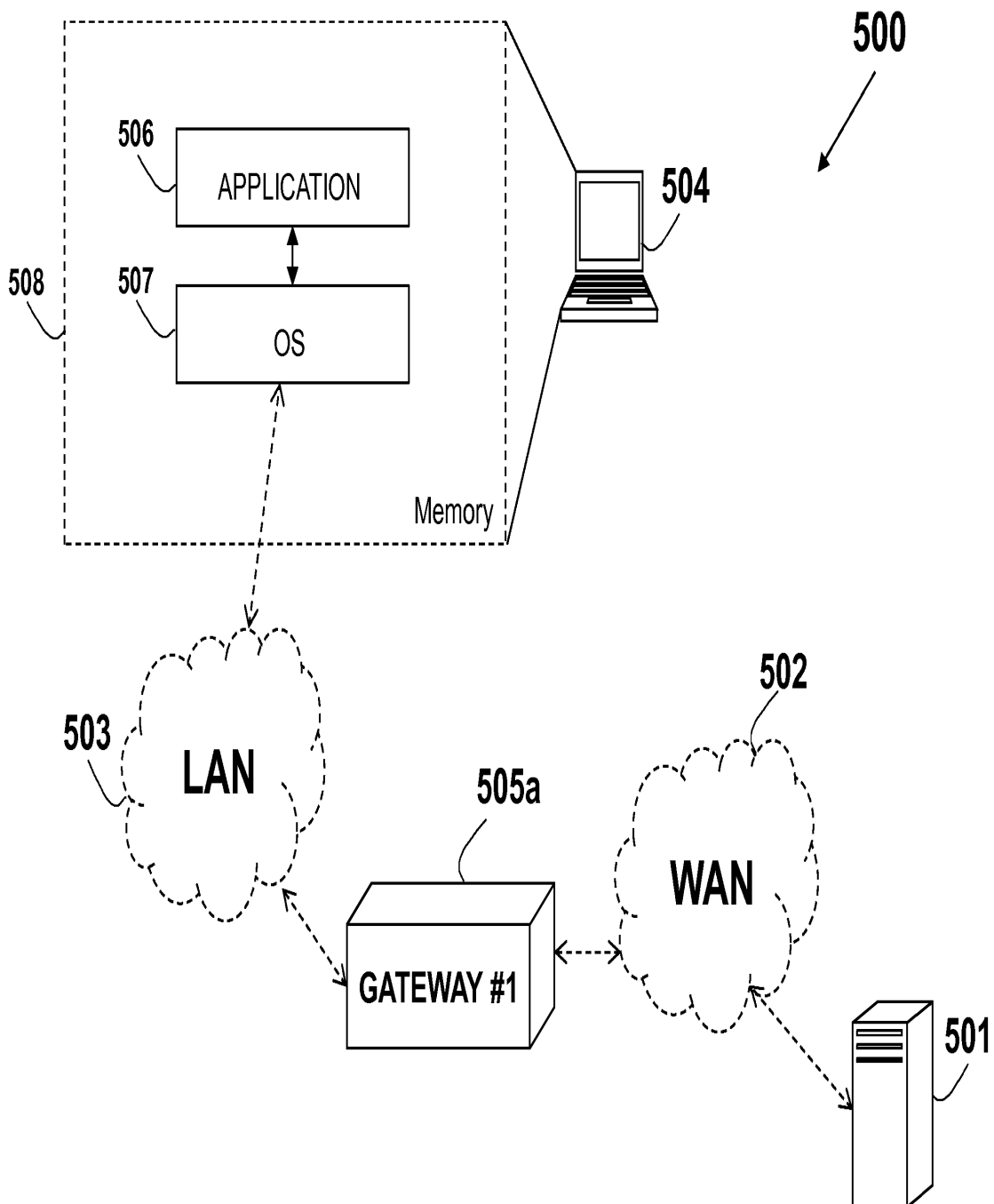


FIG. 50 (Prior Art)

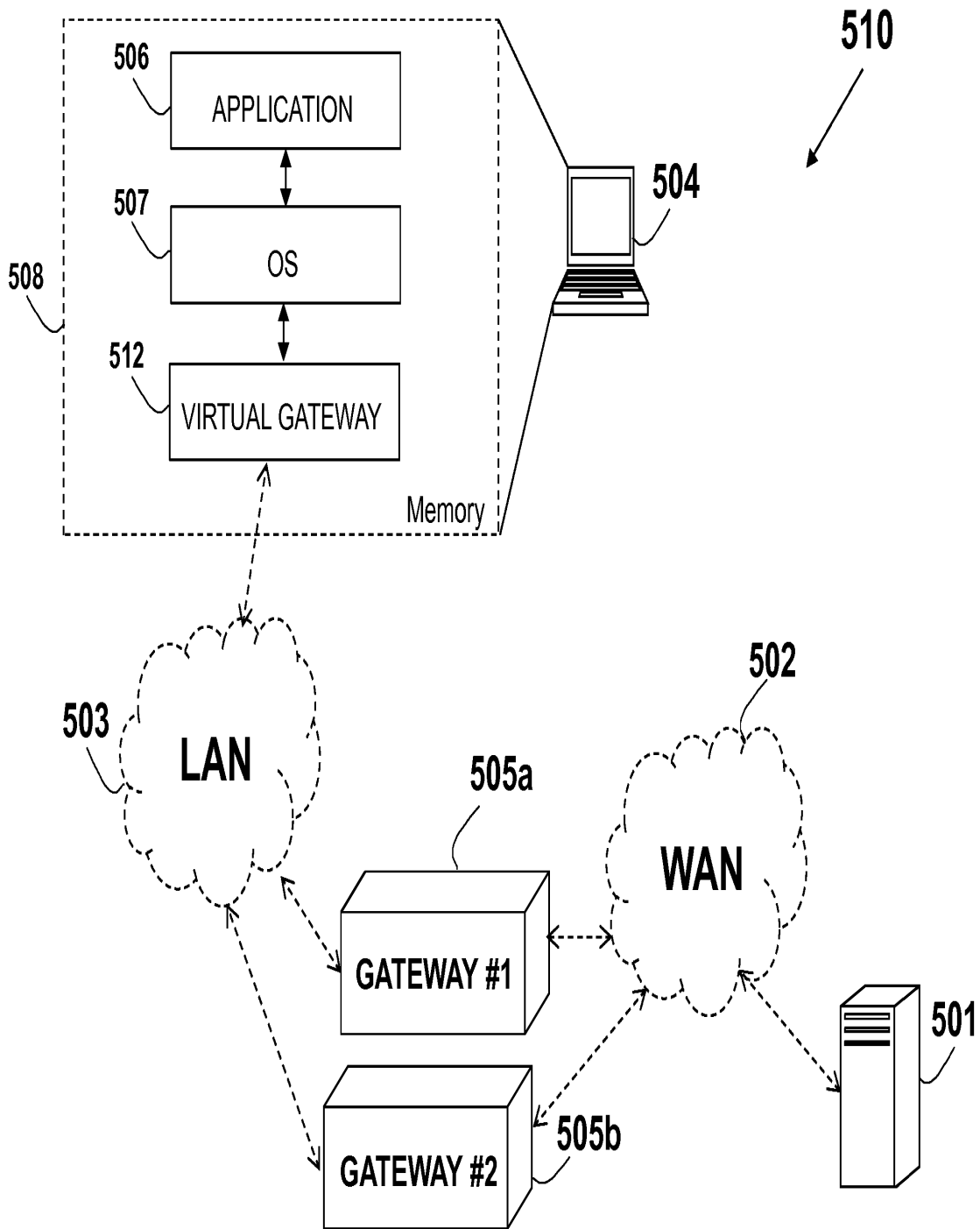


FIG. 51

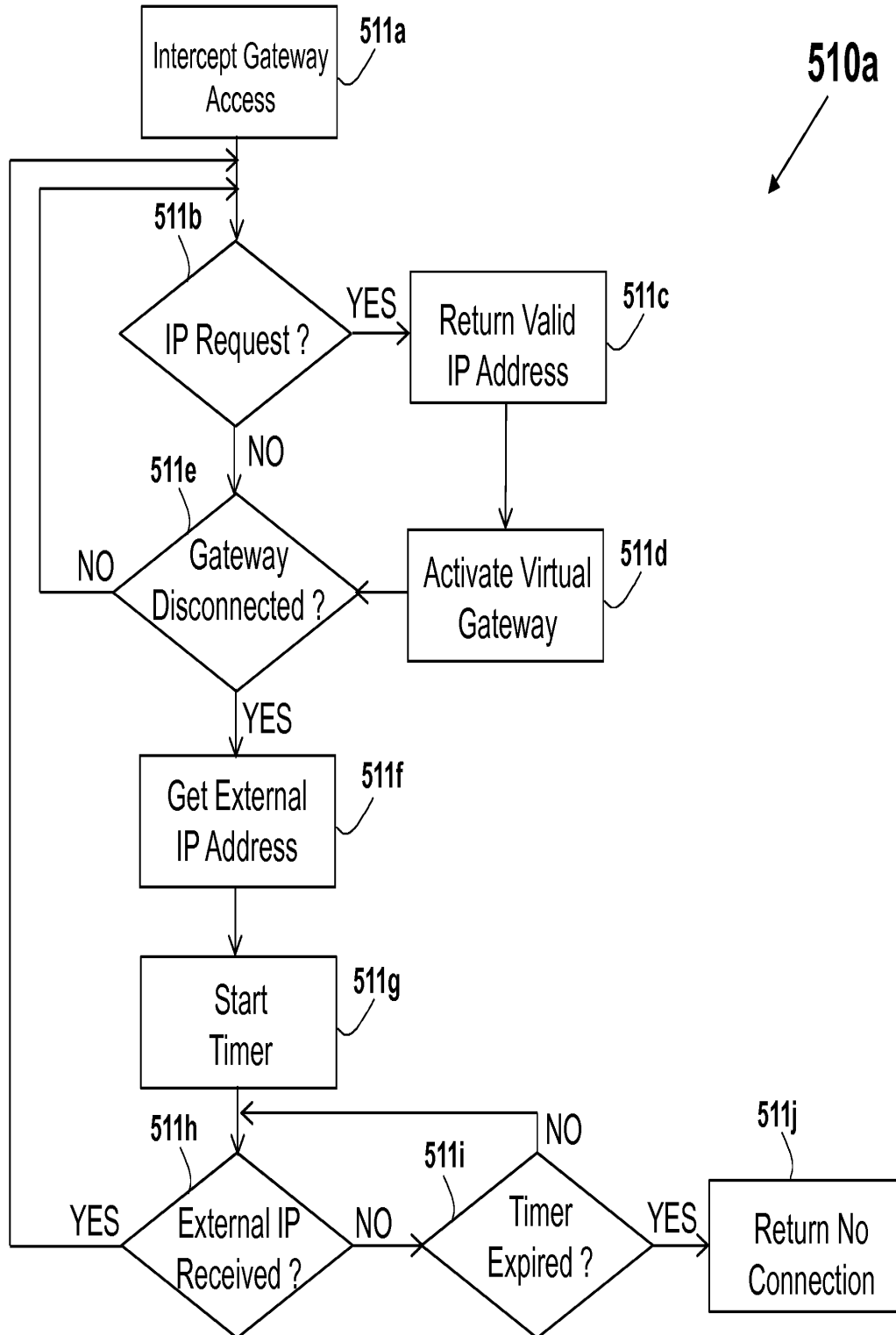
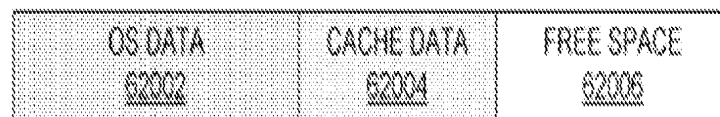


FIG. 51a

STORAGE DEVICE:



FILE SYSTEM WRITE
DIRECTION
62020



CACHE SYSTEM WRITE
DIRECTION
62030

FIG. 52

STORAGE DEVICE:



FILE SYSTEM WRITE
DIRECTION
62040



CACHE SYSTEM WRITE
DIRECTION
62050

FIG. 53

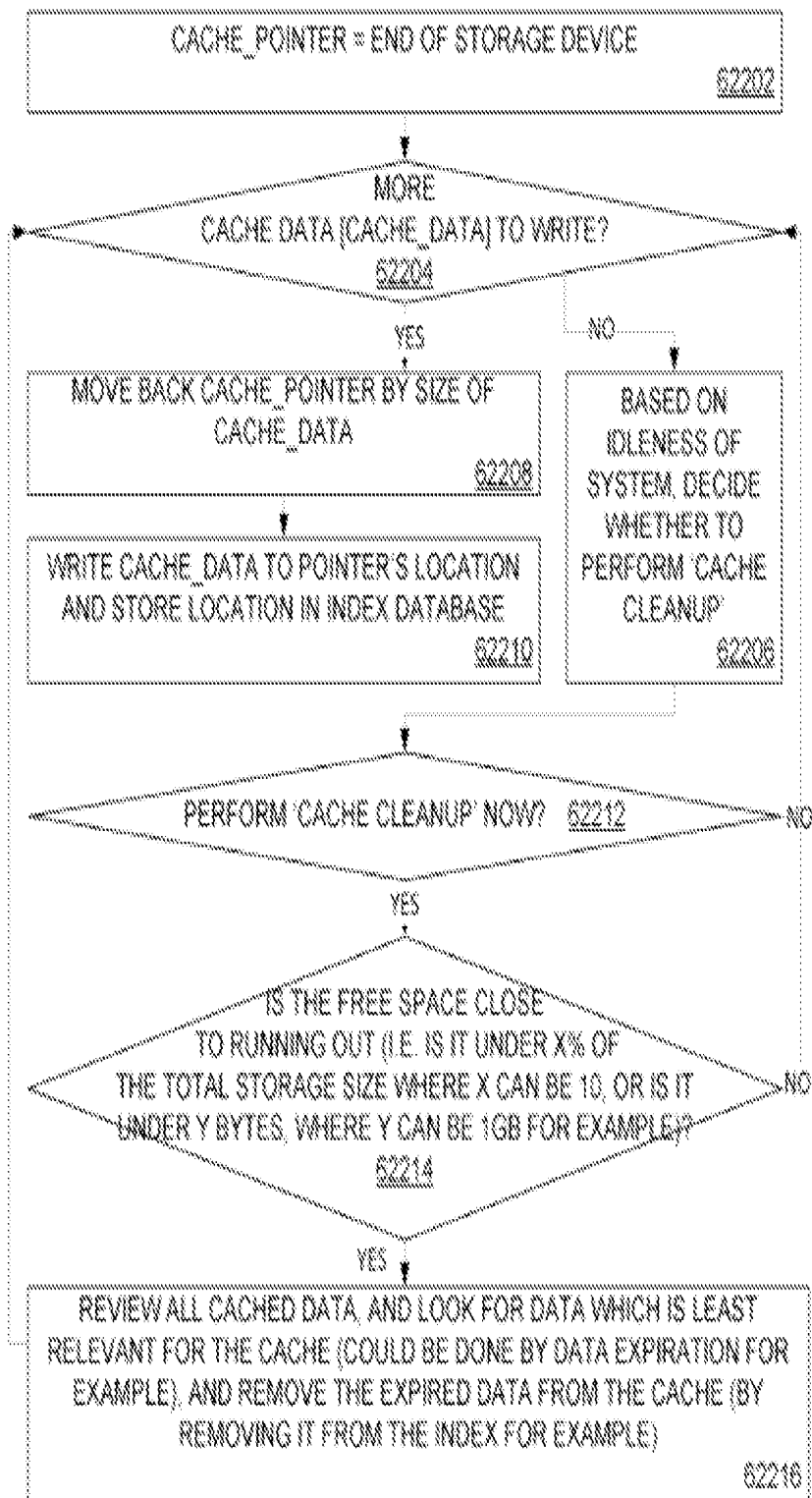


FIG. 54

STORAGE DEVICE:

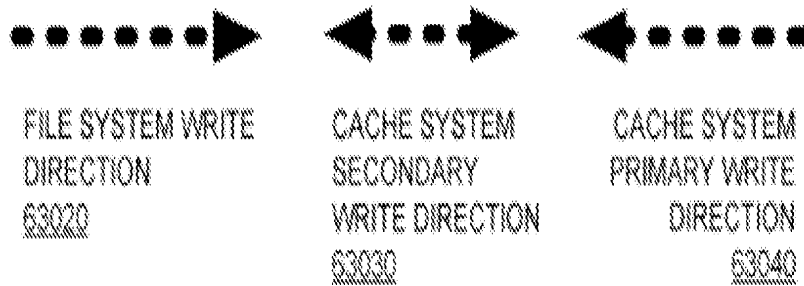
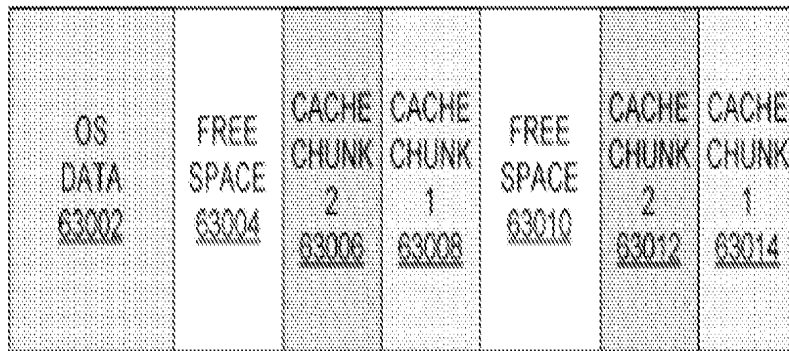


FIG. 55

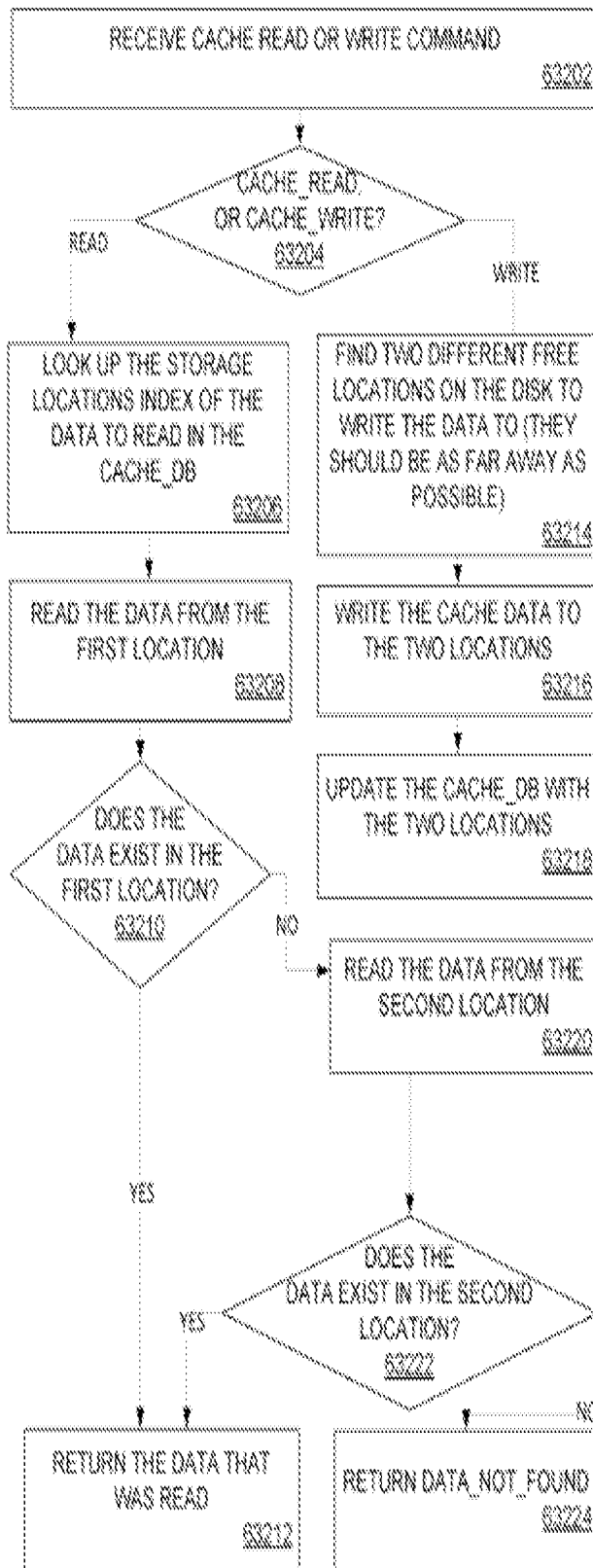
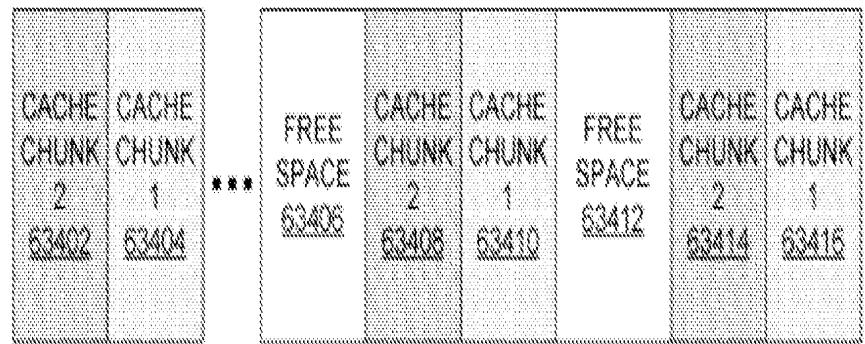


FIG. 56

STORAGE DEVICE



NTH COPY OF
CHUNKS
63420



2ND COPY OF
CHUNKS
63430



1ST COPY OF
CHUNKS
63440

FIG. 57

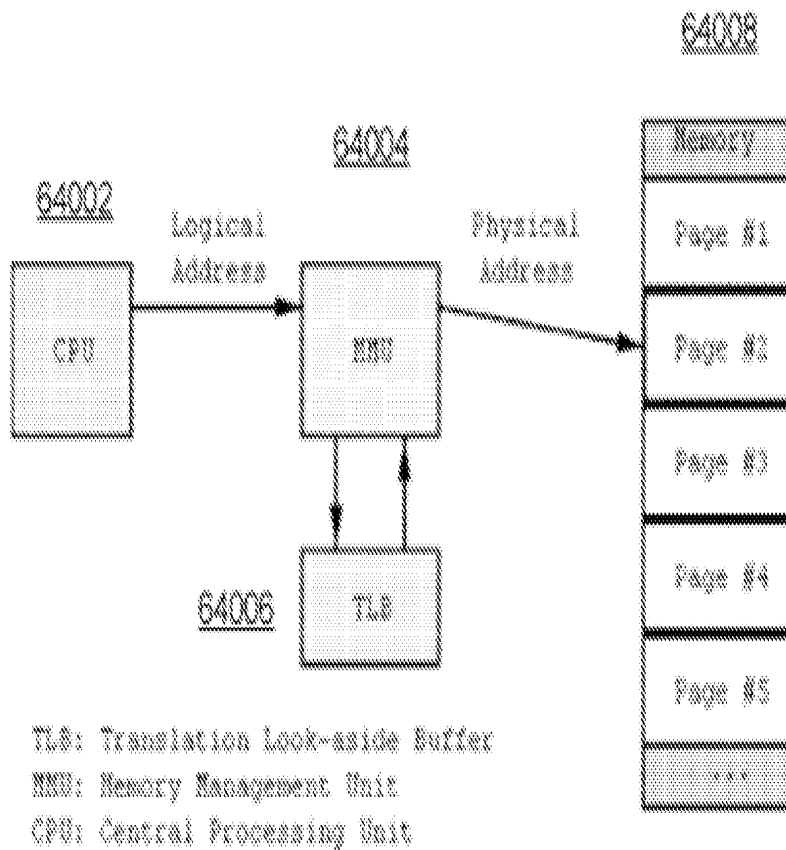


FIG. 59

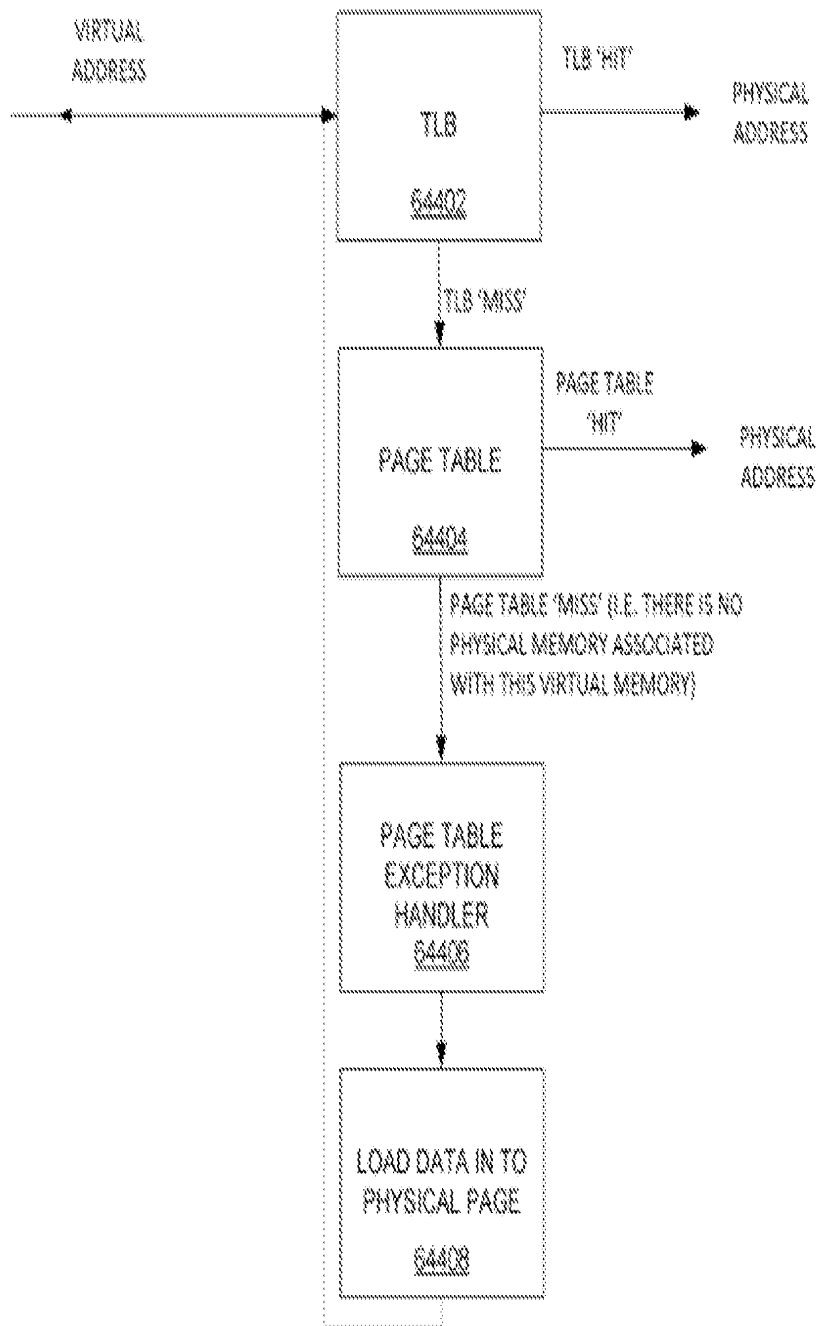


FIG. 60

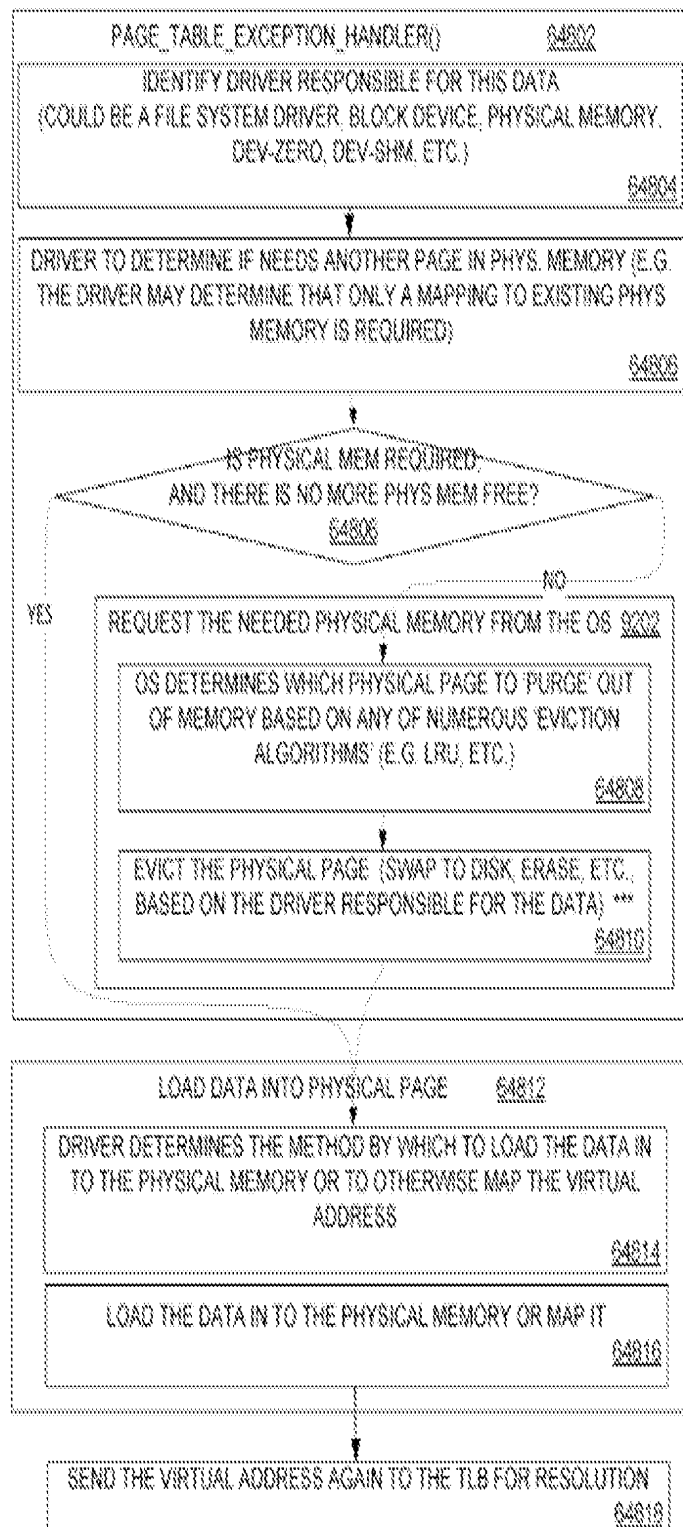


FIG. 61

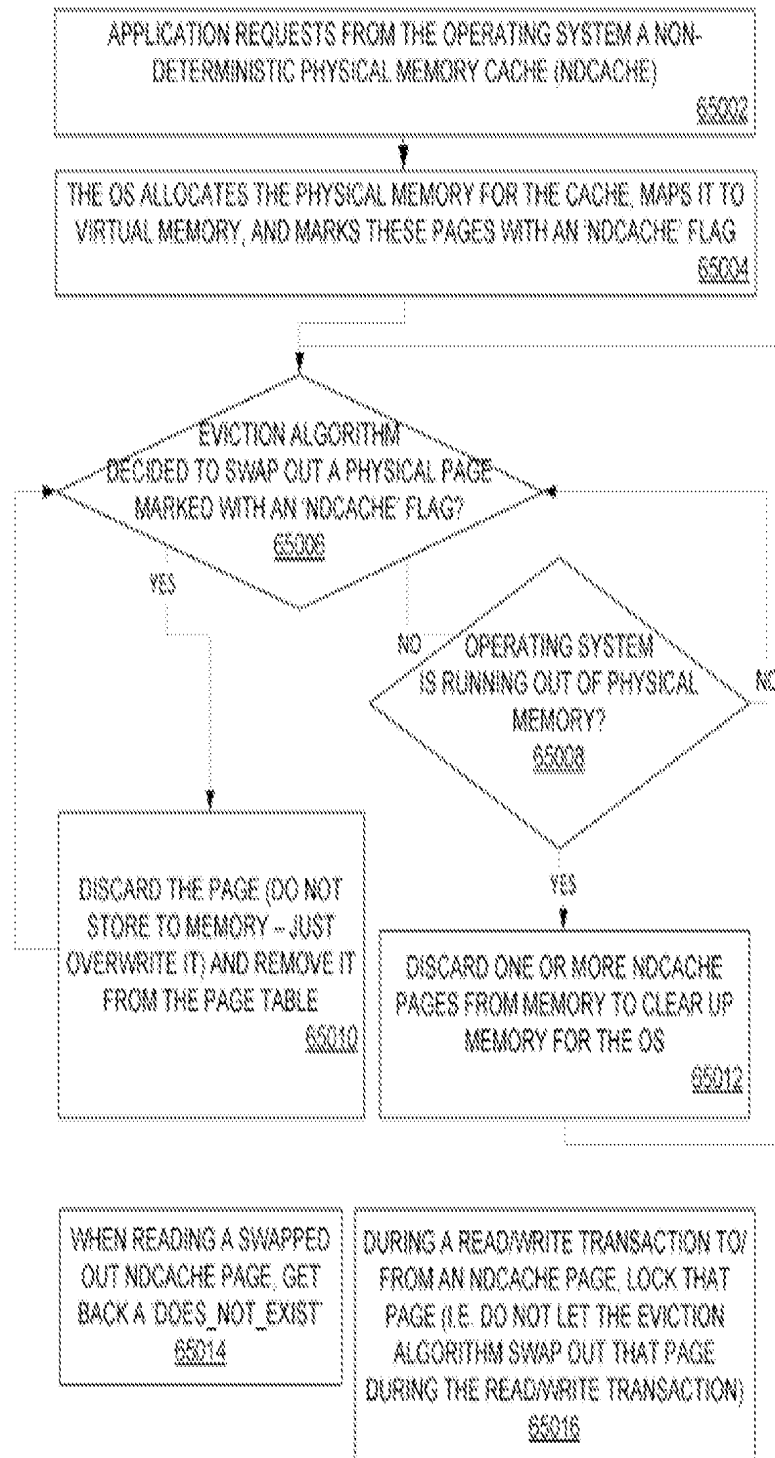


FIG. 62

API (SIMILAR TO THE FILE API IN POSIX)

FILES_NDCACHE_OPEN("NAME", CREATE_FLAG, TRUNCATE_FLAG)

- DESCRIPTION: OPEN AN NDCACHE CALLED 'NAME' FOR READING OR WRITING AND RECEIVE A HANDLE [FILES] TO IT.
- NAME - THE NAME OF THE NDCACHE TO OPEN OR CREATE
THIS NAME CAN BE USED BETWEEN DIFFERENT PROCESSES ACCESSING THIS PAGE. SIMILAR TO HOW A FILENAME IS USED
- CREATE_FLAG - IS TURNED ON TO CREATE THE CACHE IF SUCH AN NDCACHE CALLED <NAME> DOES NOT EXIST
- TRUNCATE_FLAG - IS TURNED ON TO ERASE THE NDCACHE CALLED <NAME> IF IT ALREADY EXISTS
- RETURN VALUE - THE NDCACHE_OPEN RETURNS A FILE DESCRIPTOR (FILES) TO BE USED AS A HANDLE FOR THE OTHER NDCACHE APIS

65402

NDCACHE_READ(FILES, BUFFER, SIZE)

NDCACHE_WRITE(FILES, BUFFER, SIZE)

- DESCRIPTION: READ OR WRITE TO THE NDCACHE REFERENCED BY FILES
- FILES - THE FILE DESCRIPTOR OF THE OPEN NDCACHE TO READ FROM OR WRITE TO. RECEIVED FROM NDCACHE_OPEN
- BUFFER - (FOR READ) THE BUFFER TO READ THE INFORMATION IN TO, (FOR WRITE) ... TO WRITE FROM
- OFFSET - THE OFFSET WITHIN THE NDCACHE TO START THE READ OR WRITE FROM
- SIZE - THE NUMBER OF BYTES TO READ OR WRITE

65404

NDCACHE_CLOSE(FILES)

- DESCRIPTION: CLOSES THE HANDLE TO THE NDCACHE DESCRIBED BY FILES. AFTER ALL HANDLES TO AN NDCACHE ARE CLOSED, THE OS MAY DECIDE TO SWAP IT OUT BASED ON ITS EVICTION ALGORITHM
- FILES - THE FILE DESCRIPTOR OF THE OPEN NDCACHE TO CLOSE

65406

NDCACHE_UNLINK(FILES)

- DESCRIPTION: REMOVE AN NDCACHE ENTRY FROM THE PAGE TABLE
- FILES - THE FILE DESCRIPTOR OF THE OPEN NDCACHE TO REMOVE FROM THE PAGE TABLE

65408

FIG. 63

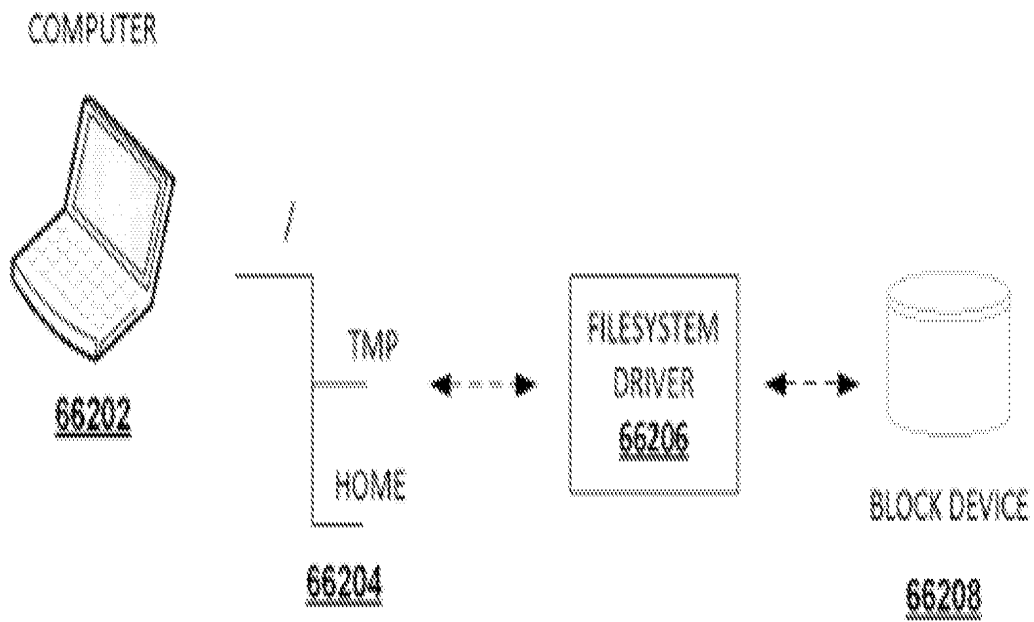


FIG. 64

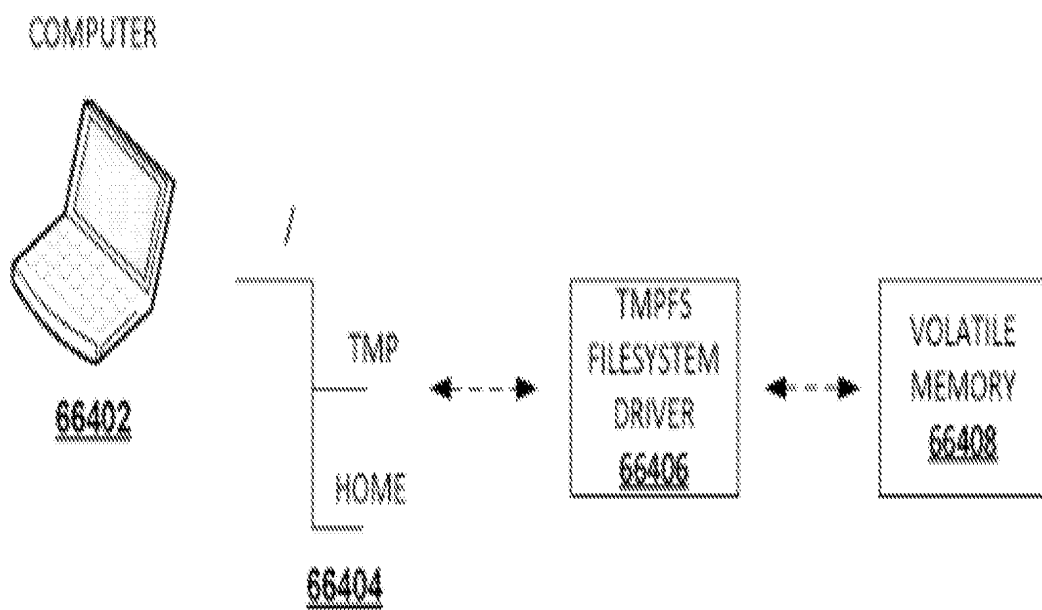


FIG. 65

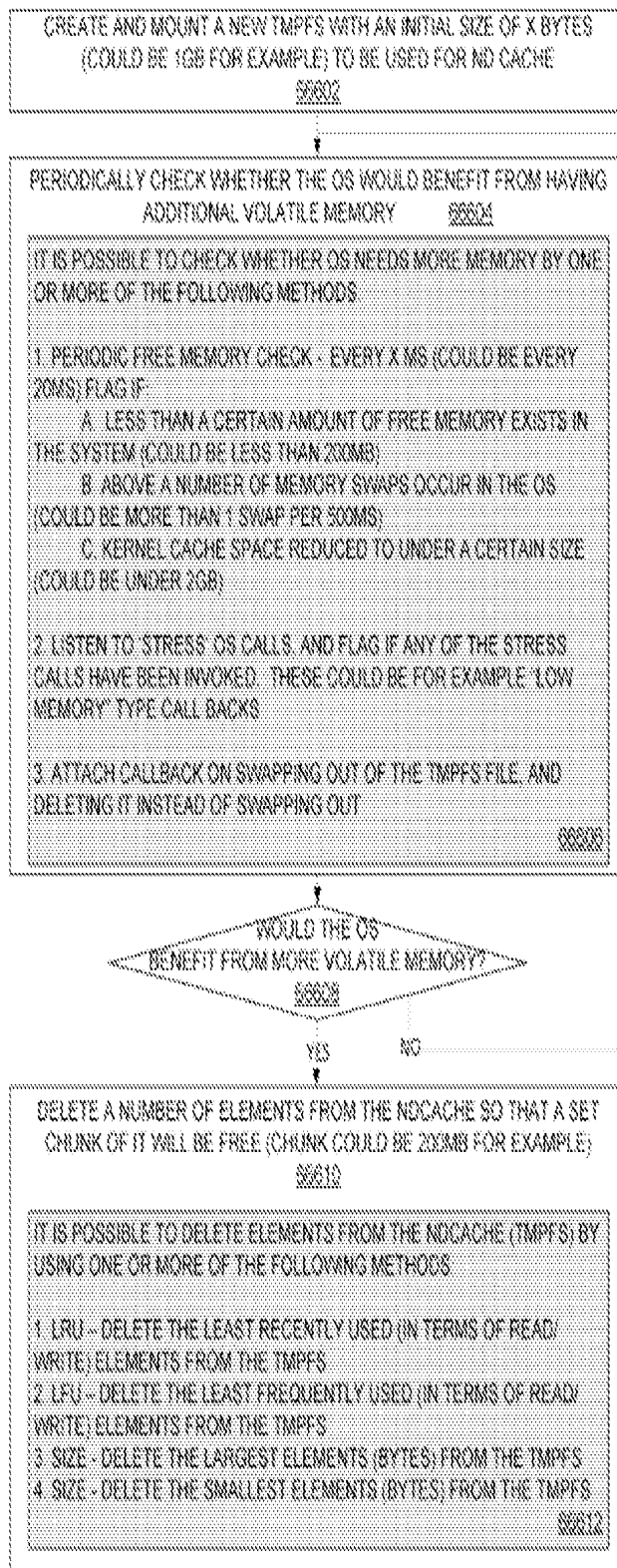


FIG. 66

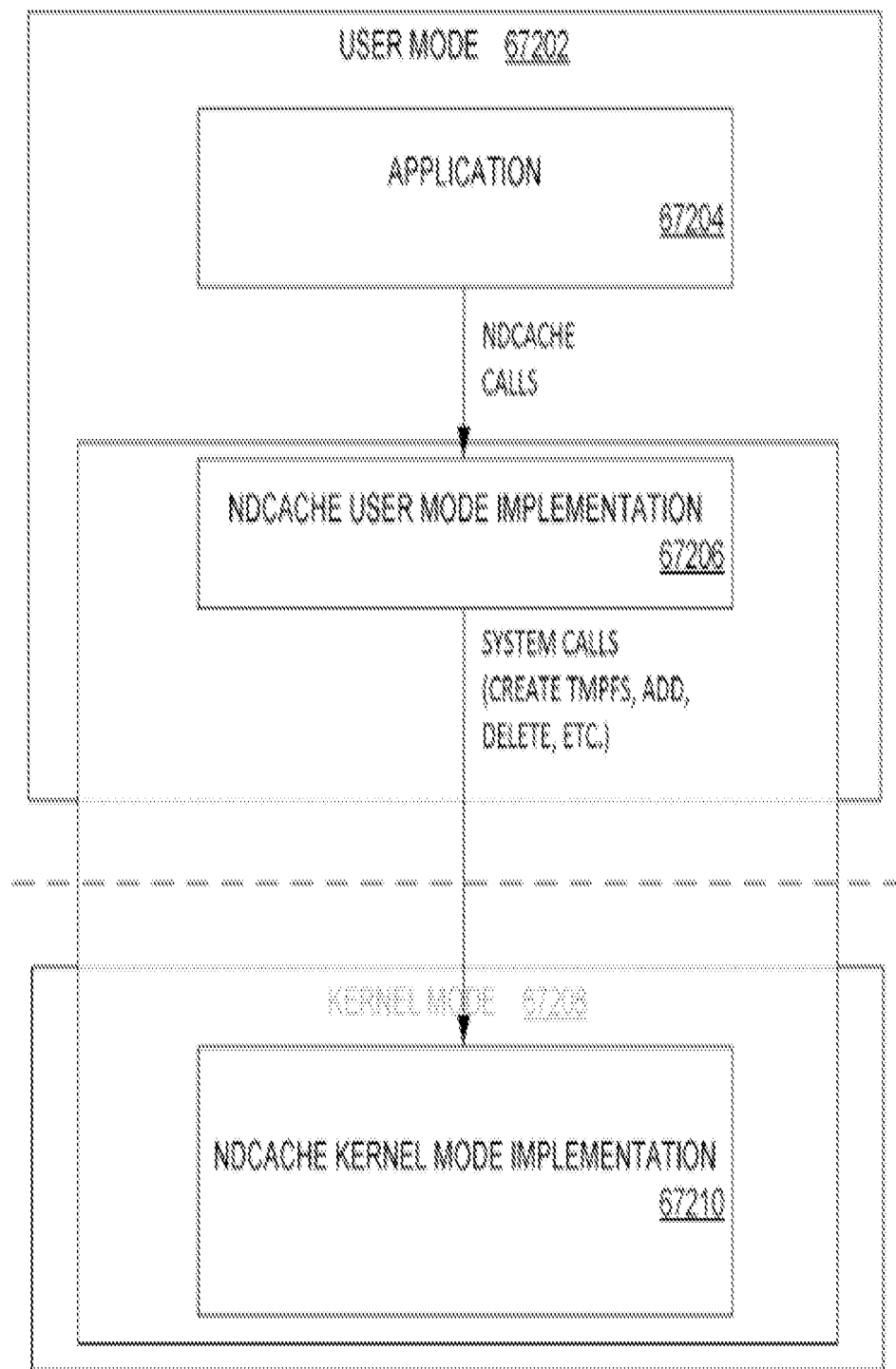


FIG. 67

WRITING TO (AND ALLOCATING) THE NDCACHE:

```
FILEDS *FILEDS=OPEN("NDCACHE_DRIVER/NDCACHE_NAME");
/* NDCACHE_DRIVER IS THE KERNEL MODULE FOR NDCACHE EVENTS
   NDCACHE_NAME IS THE NAME OF THIS NDCACHE, TO BE SHARED
   BETWEEN DIFFERENT PROCESSES */

INT *P=NDCACHE_LOOKUP(NDCACHE_NAME); // GET PTR TO THE NDCACHE

IF (!P)
{
    // NO SUCH NDCACHE EXISTS, AND THUS NEEDS TO BE CREATED

    INT *P = MMAP(1MB, SHARED MEMORY, FILEDES);
    // 1MB AS AN EXMAPLE NDCACHE MEM SIZE REQUESTED
    // SHARED MEMORY SO THAT CAN BE USED BY MULTIPLE PROCESSES

    (INT* P)[0]++; // INDICATES THAT THERE IS CONTENT IN THE NDCACHE
    P[N]=...; // WRITE THE NEW CONTENT TO THE NDCACHE
}
ELSE
    DO NDCACHE_WRITE(); // THIS NDCACHE EXISTS, WRITE TO IT
```

67402

FIG. 68

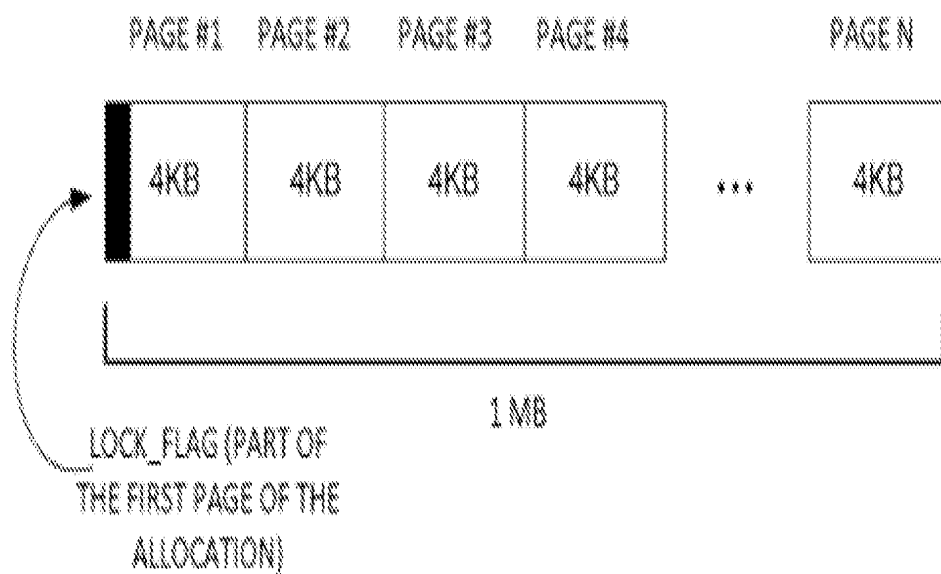


FIG. 69

NDCACHE_READ()/NDCACHE_WRITE():

```
// CHECK IF THE NDCACHE STILL EXISTS (IF IT WAS SWAPPED OUT BY THE
// OS, IT NO LONGER EXISTS)
IF (!P[0]) RETURN MEM_NOT_FOUND;

// COULD BE THAT AT THIS POINT THE OS SWAPPED OUT THIS NDCACH, SO
// CHECK IF THE NDCACHE STILL EXISTS, AND LOCK IT FOR THE DURATION
// OF THE READ/WRITE
IF ((++P[0])==1)
{
    P[0]--;
    RETURN MEM_NOT_FOUND;
}

// READ/WRITE CONTENT FROM/TO THE NDCACHE
BUF = P[N]; // FOR READ
OR
P[N] = ...; // FOR WRITE

// DECREMENT THE LOCK_FLAG BY ONE (LOCK_FLAG--), INDICATING THAT
// THE NDCACHE LOCK FOR THIS READ/WRITE IS RELEASED
P[0]--;

RETURN;
```

67802

FIG. 70

NDCACHE KERNEL MODULE (NDCACHE_DRIVER)

**EVICTON ALGORITHM – WHEN THE OS NEEDS MORE PHYSICAL MEMORY
AND IS CONSIDERING WHICH PHYSICAL MEMORY TO EVICT:**

FOR EACH OF THE MEMORY RANGES HANDLED BY THE NDCACHE DRIVER,
WHERE LOCK_FLAG=MEM_RANGE[0].

IF LOCK_FLAG == 0 // MEM RANGE NOT IN USE
THEN SWAP OUT THIS MEMORY RANGE

IF LOCK_FLAG == 1 // NDCACHE MEM RANGE IS IN USE, BUT NOT LOCKED
THEN PREFER NOT TO SWAP OUT THIS MEMORY RANGE

IF LOCK_FLAG > 1 // NDCACHE MEM RANGE IS IN USE AND LOCKED
DON'T SWAP OUT THIS MEMORY RANGE

67802

**ON SWAP OF THE NDCACHE MEMORY (I.E. WHEN THE OS'S EVICTION
ALGORITHM DECIDES TO SWAP OUT THE NDCACHE'S PHYSICAL MEMORY
ALLOCATION):**

MAP THE VIRTUAL MEMORY POINTERS TO AN 'ALL ZERO' RANGE IN READ
ONLY MODE (FOR EXAMPLE BY MAPPING TO DEVZERO)

67804

FIG. 71

FOURTH METHOD – ONE MANAGEMENT AREA PER PAGE:

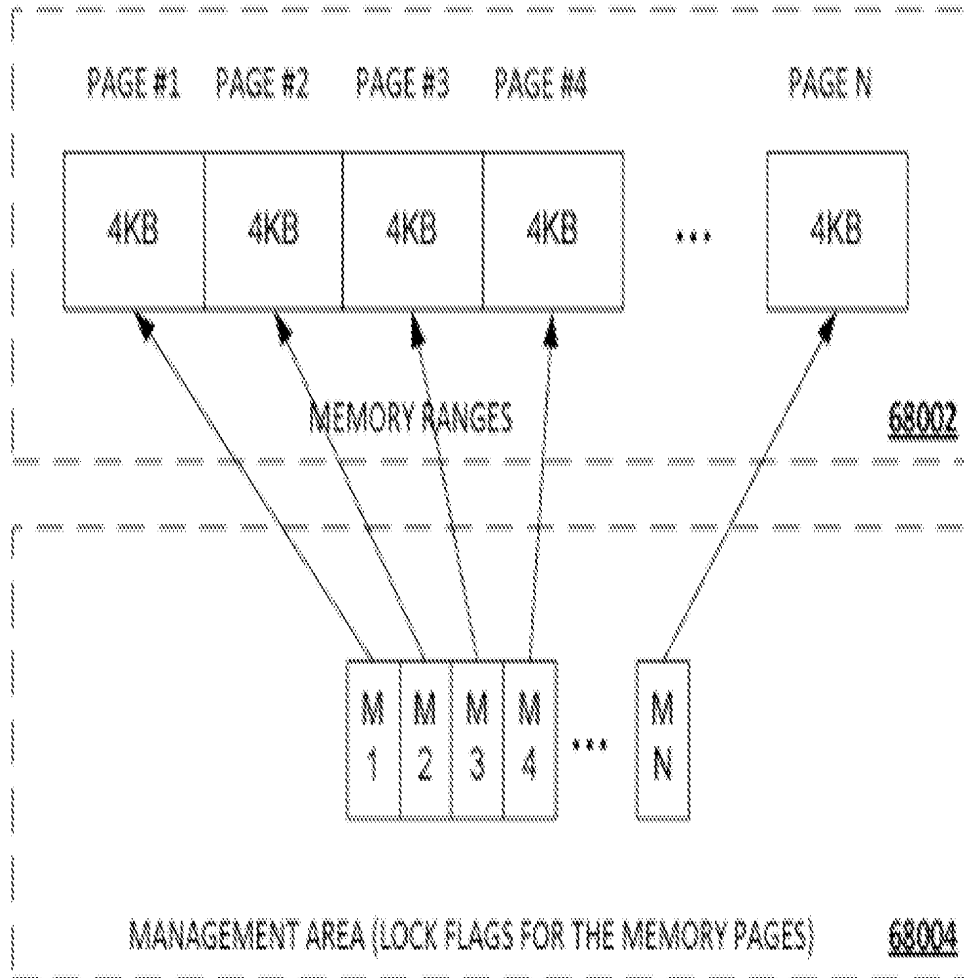


FIG. 72

INITIALIZATION:

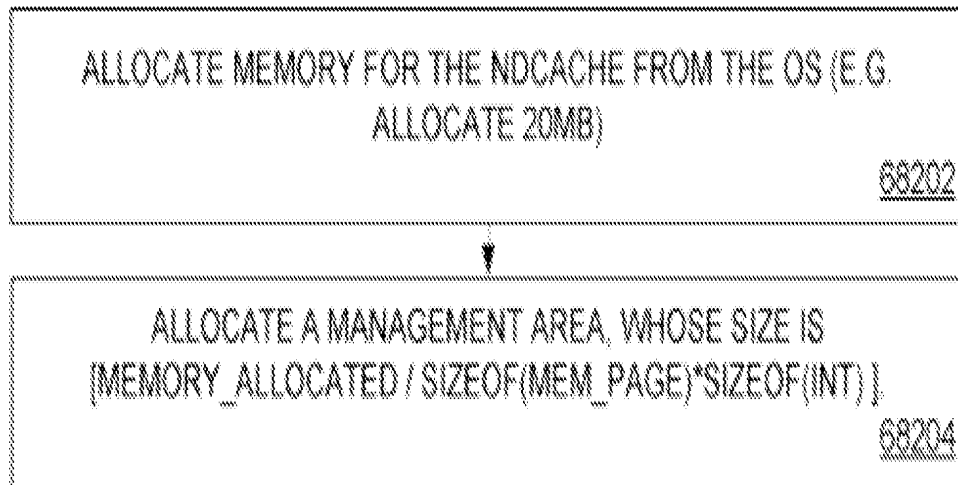


FIG. 73

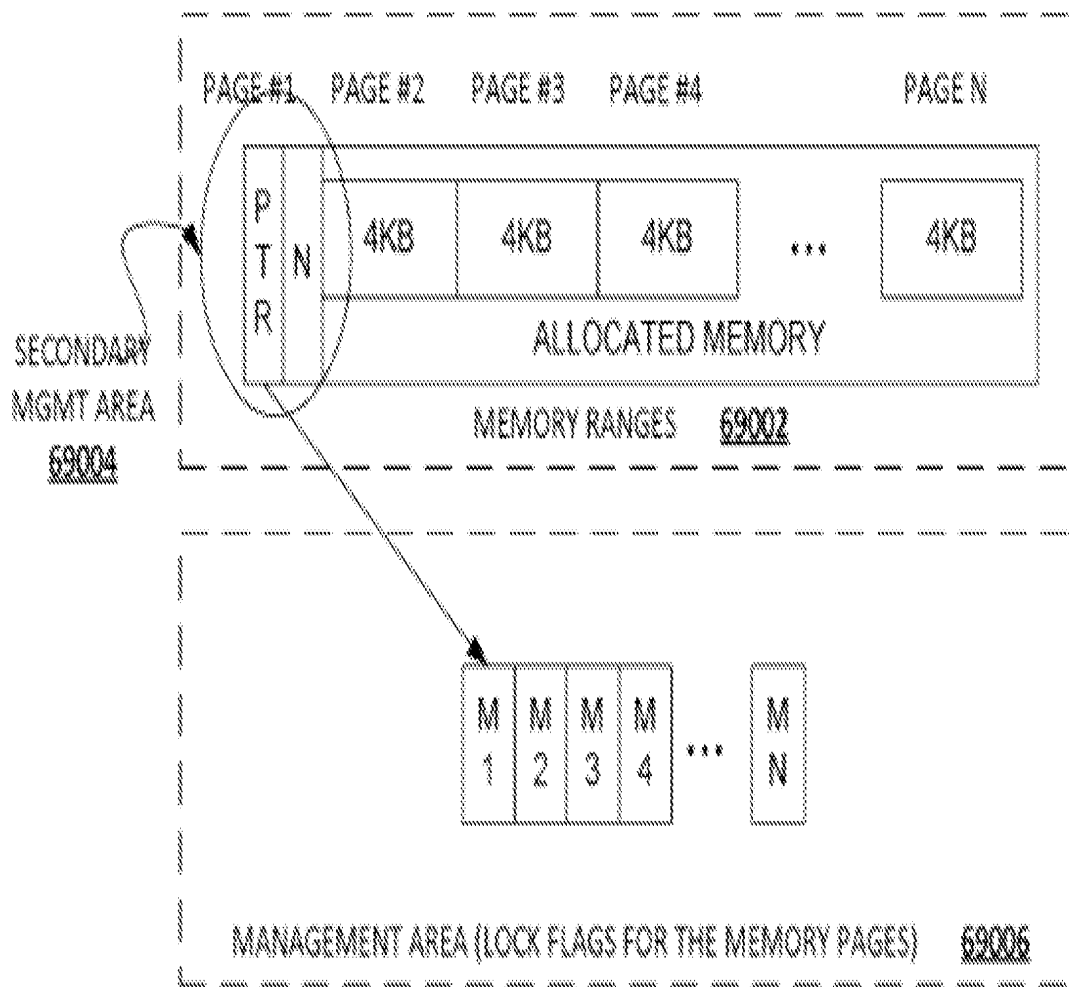


FIG. 74

NEW NDCACHE ELEMENT ALLOCATION: $P = \text{NDCACHE_CREATE}(\text{SIZE})$:

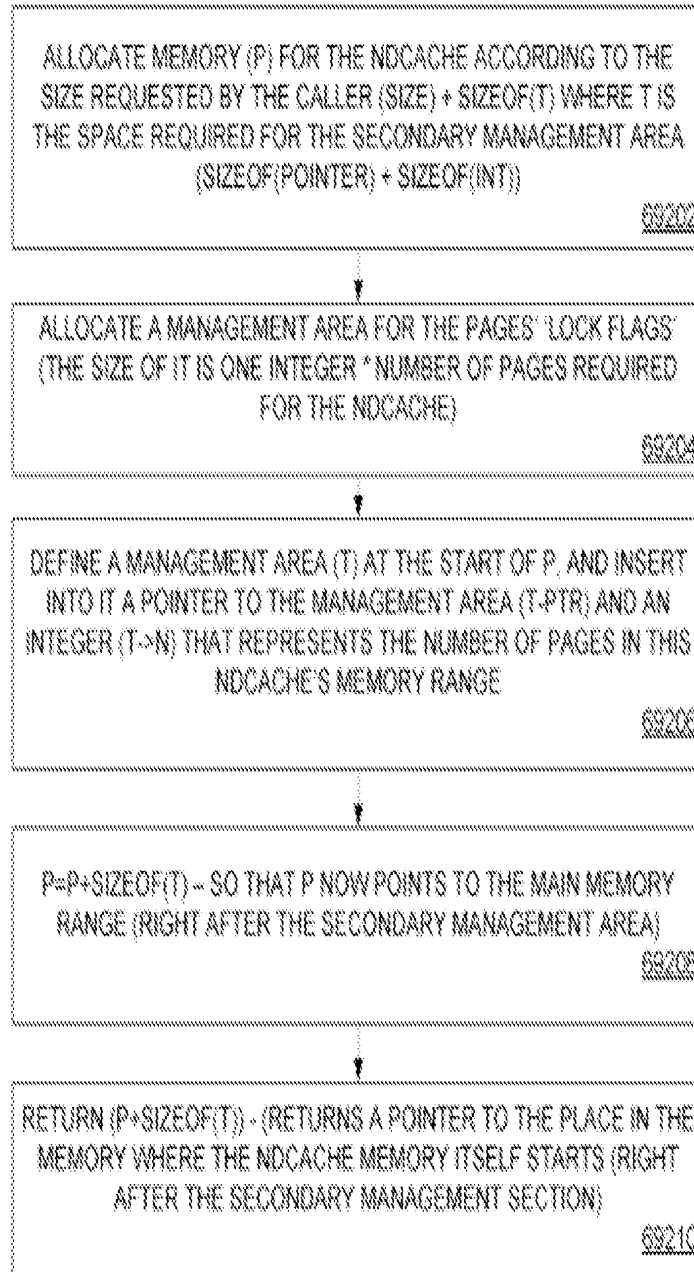


FIG. 75

READING FROM OR WRITING TO THE NDCACHE ELEMENT:

USE THE SAME ALGORITHM AS IN THE FOURTH IMPLEMENTATION
WITH TWO EXCEPTIONS:

1. LOCK(P) / UNLOCK(P) –
 - A. FIRST EVALUATES THAT T->N IS NOT ZERO. IF IT IS, THEN THE NDCACHE WAS SWAPPED AND NEEDS TO RETURN MEM_NOT_FOUND
 - B. IF T->N IS NOT ZERO, LOCK ALL PAGES THAT THIS NDCACHE ELEMENT SPANS BY LOCKING THE N LOCK_FLAGS STARTING AT THE T->PTR LOCK_FLAG
2. WHEN READING OR WRITING, NOT LIMITED TO ONE PAGE OF MEMORY (CAN USE THE ALLOCATED SIZE)

69602

DELETING THE ALLOCATION: NDCACHE_CLOSE (P)

FREE (P - SIZEOF(T)) – FREES THE SECONDARY MANAGEMENT AREA
AS WELL AS THE MAIN MEMORY AREA ALLOCATED TO THIS
NDCACHE ELEMENT

69606

FIG. 76

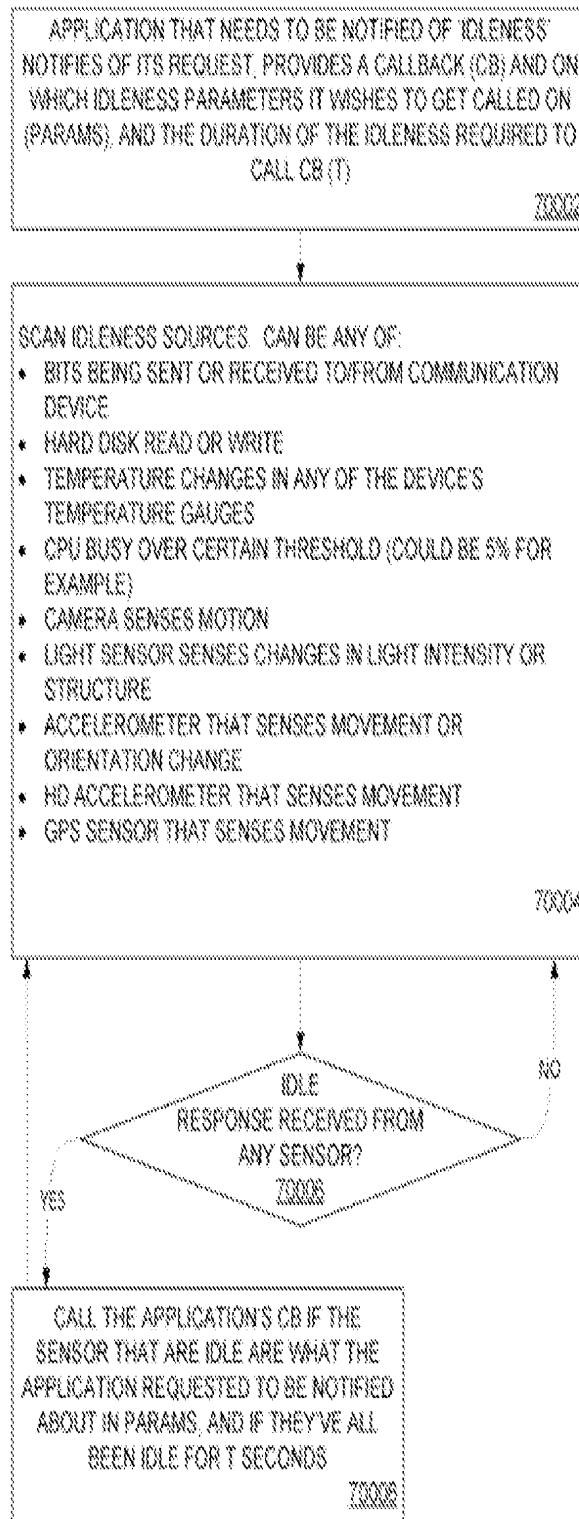


FIG. 77

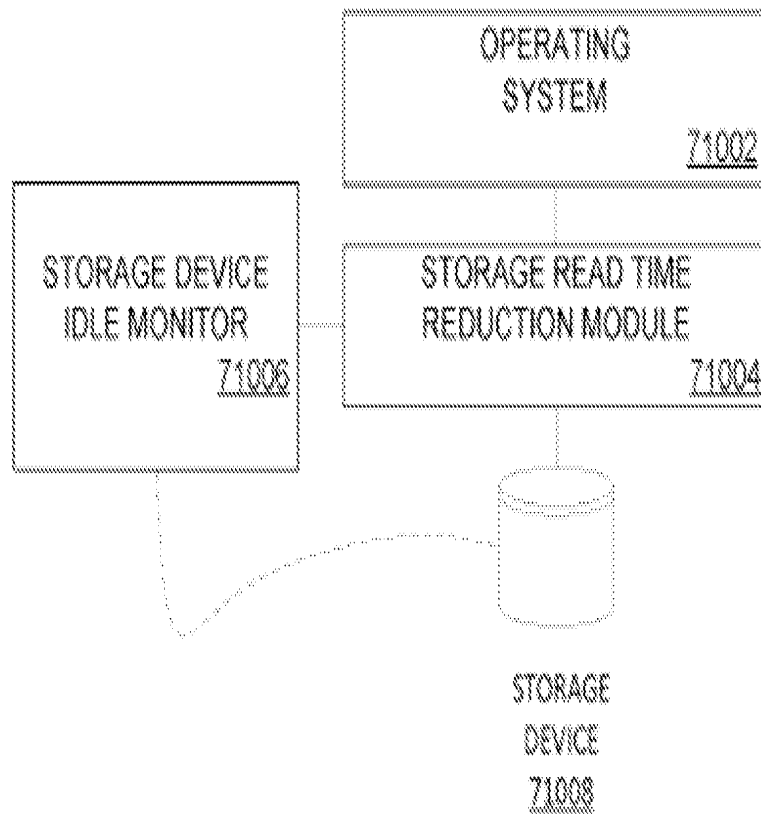


FIG. 78

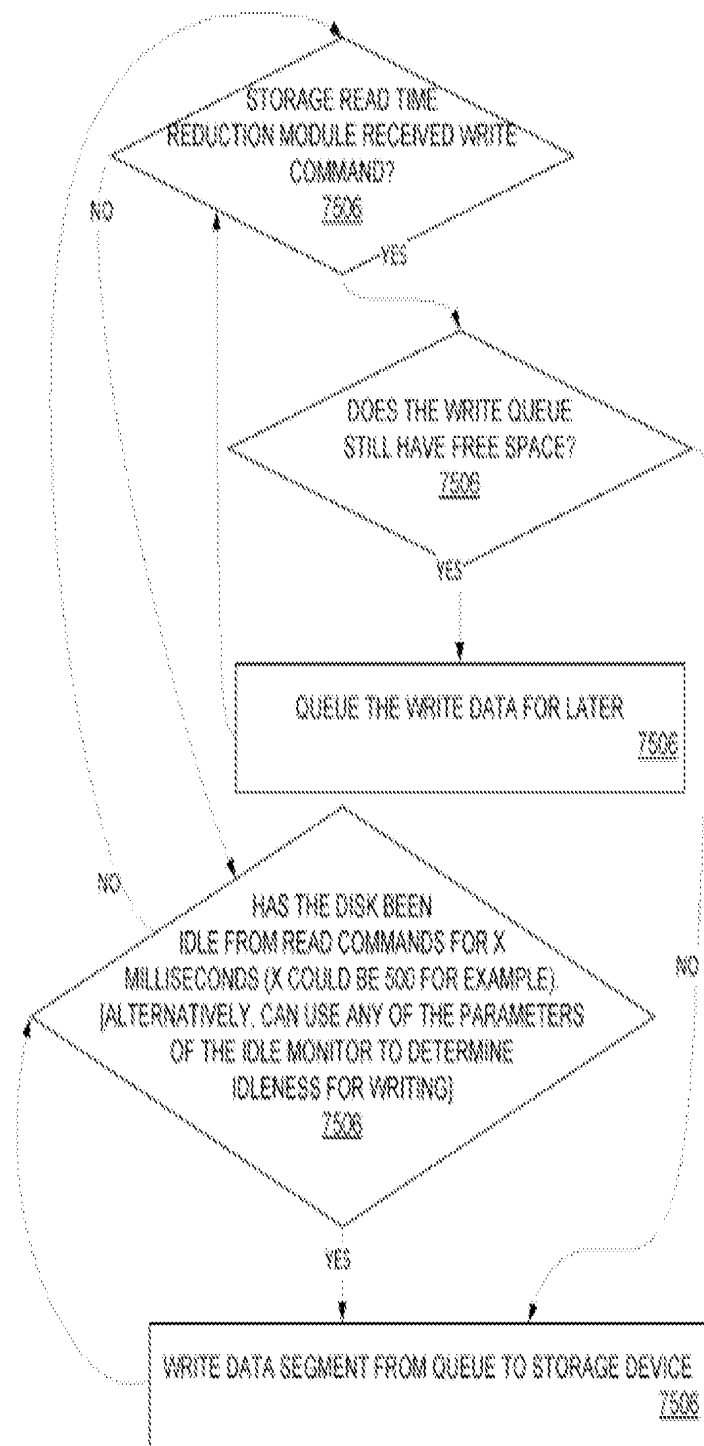


FIG. 79

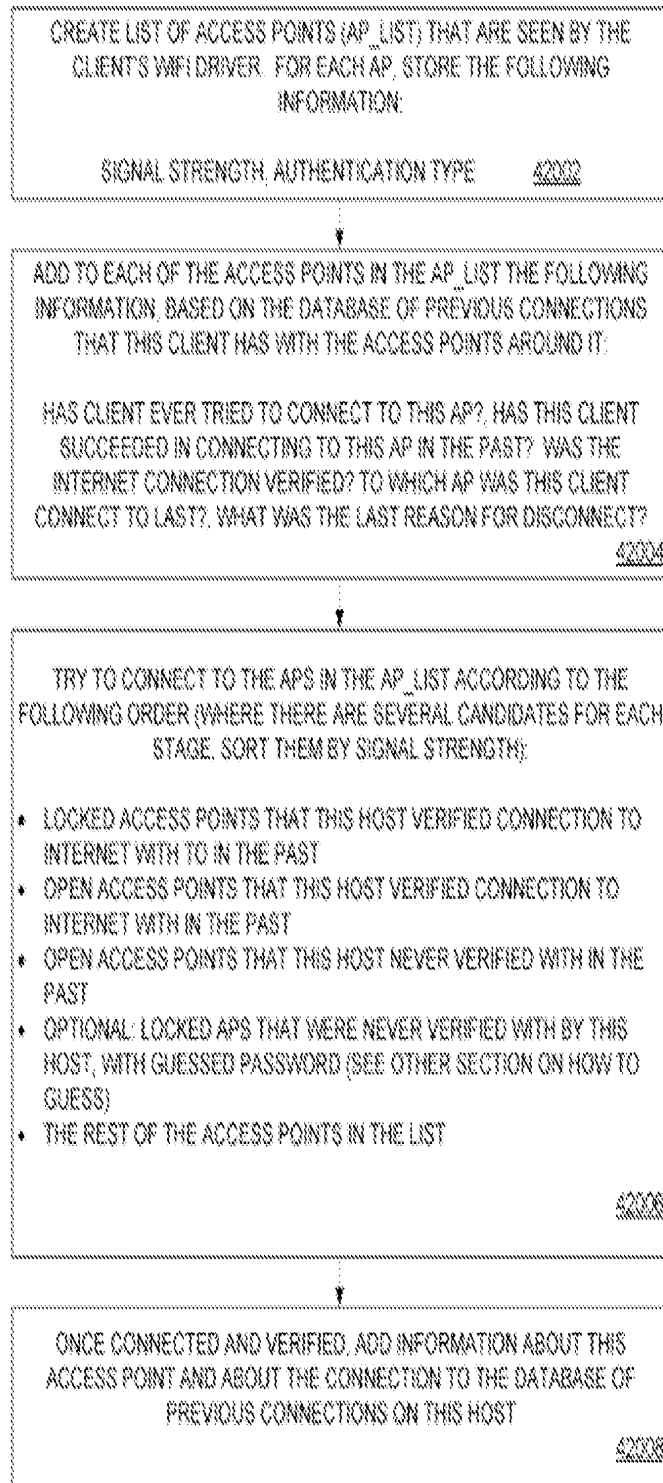


FIG. 80

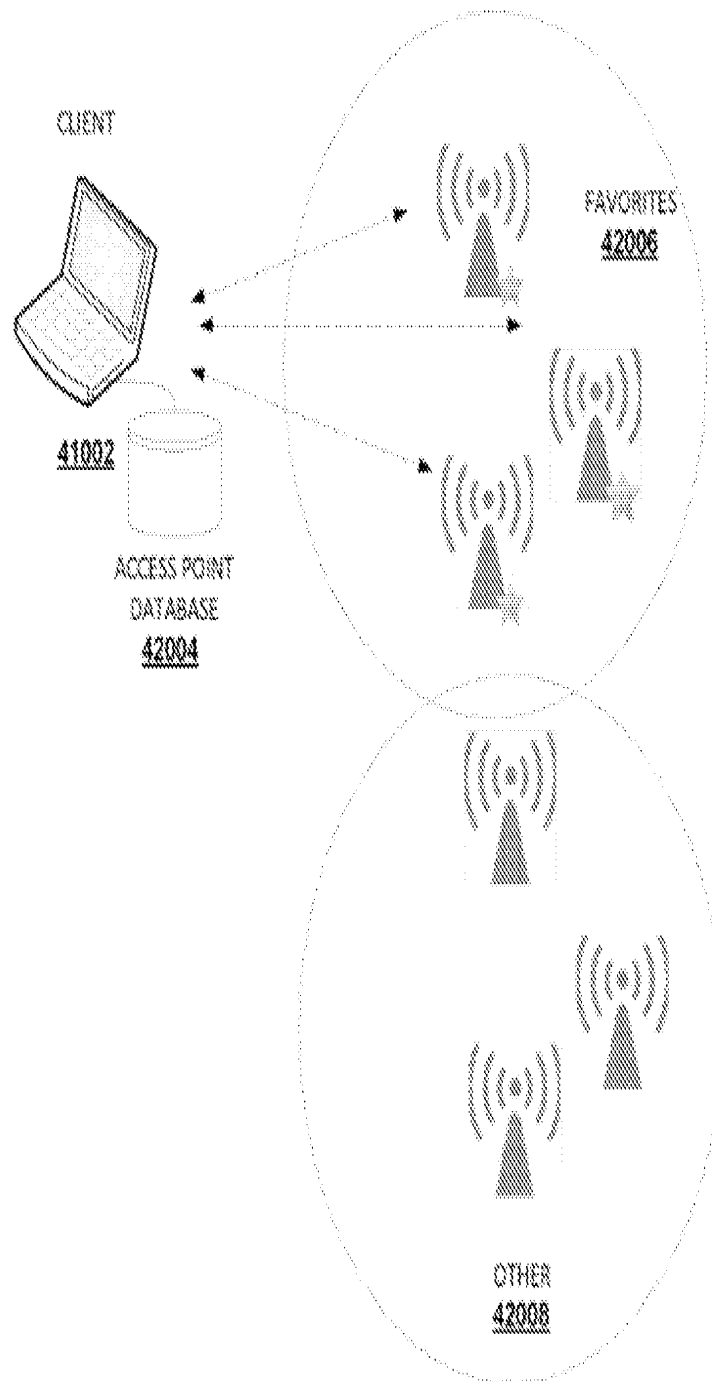


FIG. 81 (Prior Art)

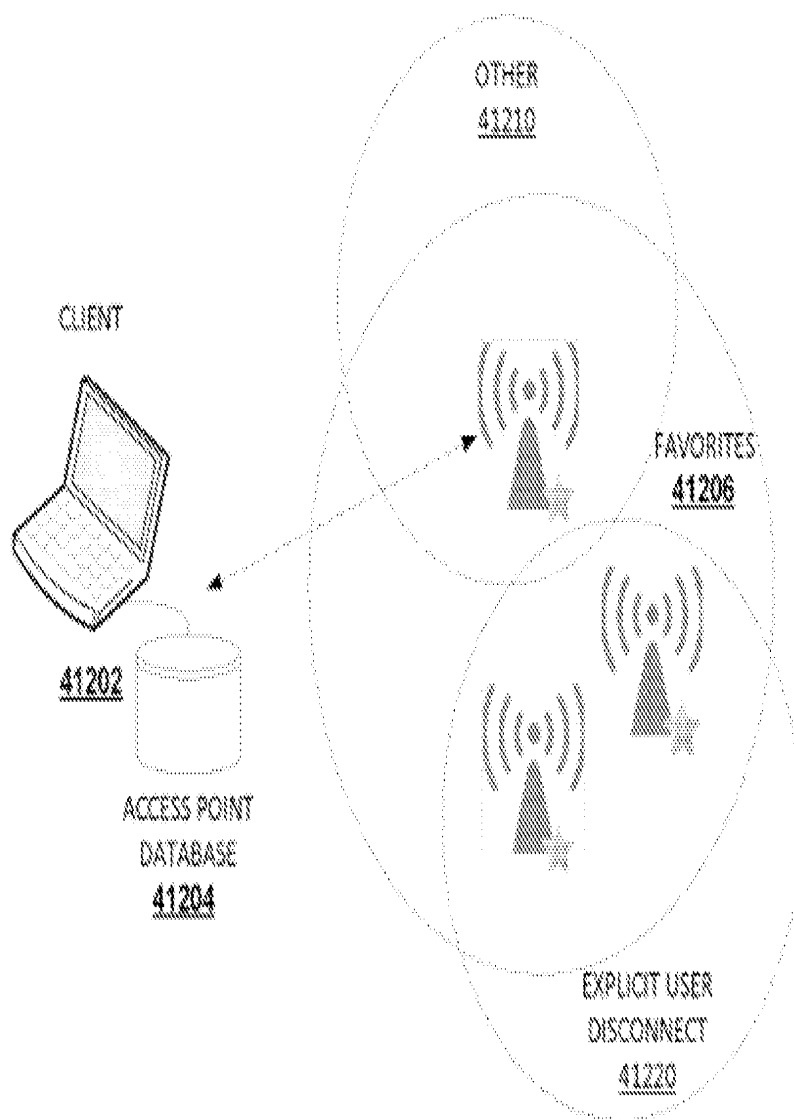


FIG. 82

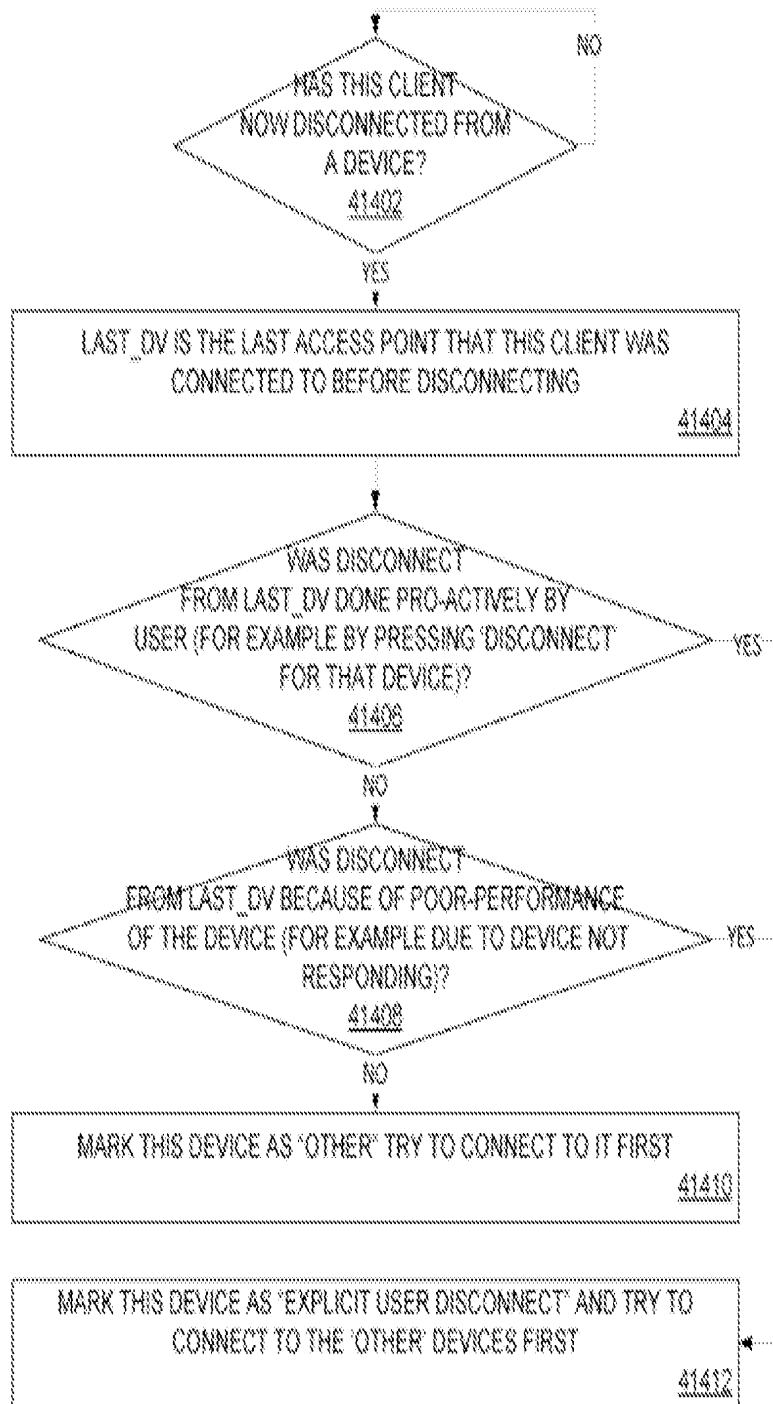


FIG. 83

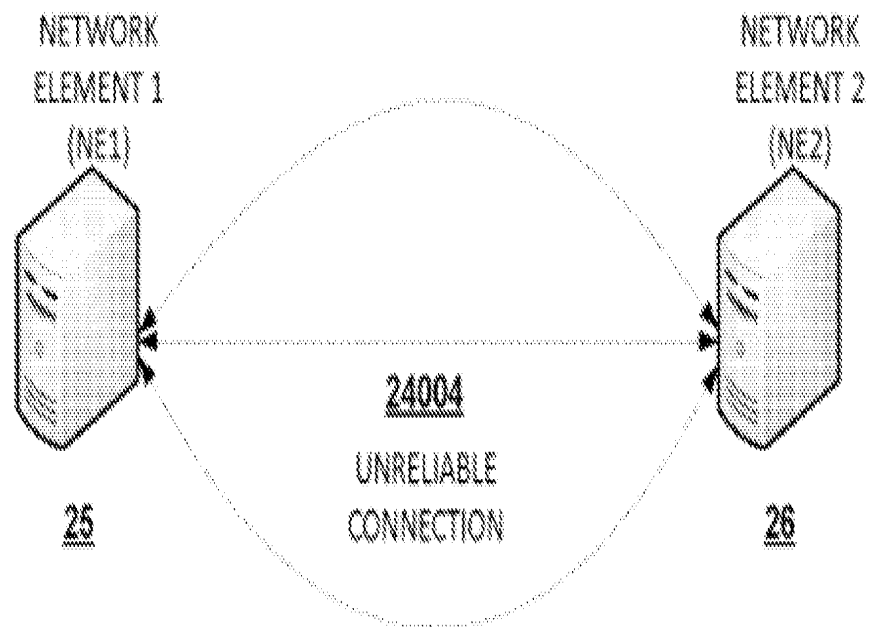


FIG. 84 (Prior Art)

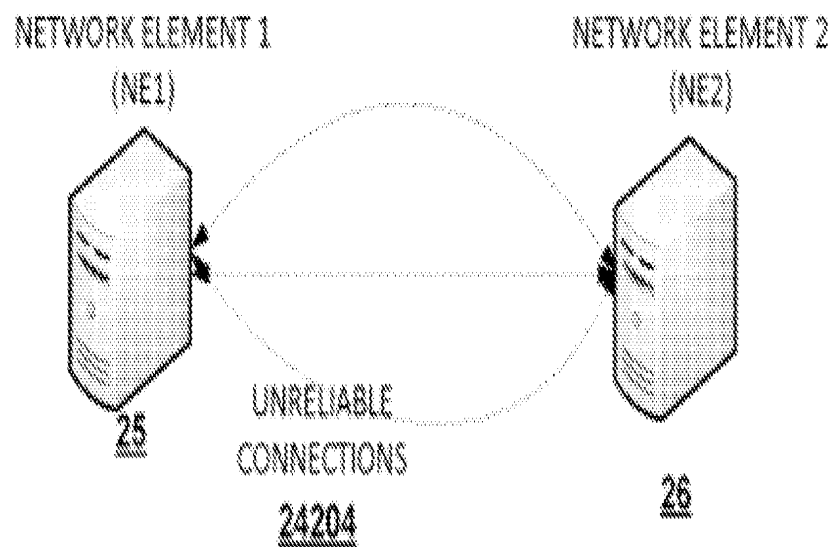


FIG. 85

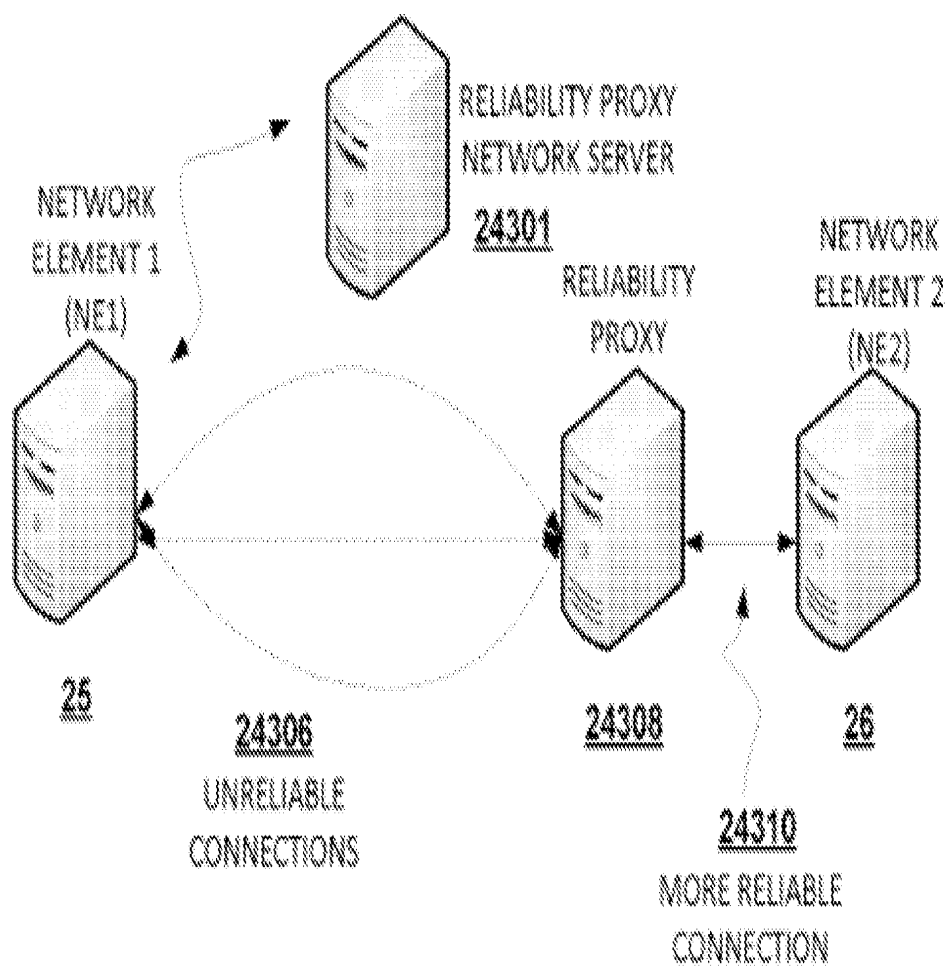


FIG. 86

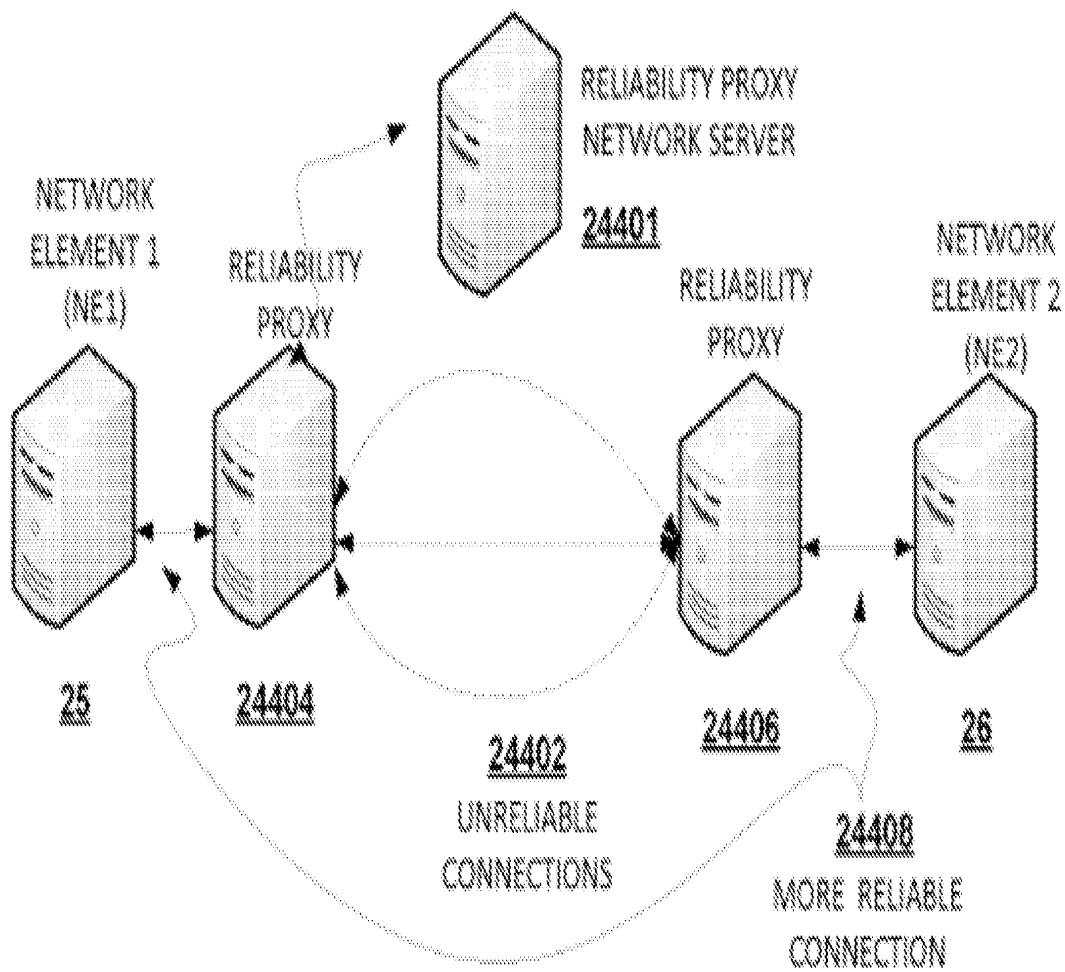


FIG. 87

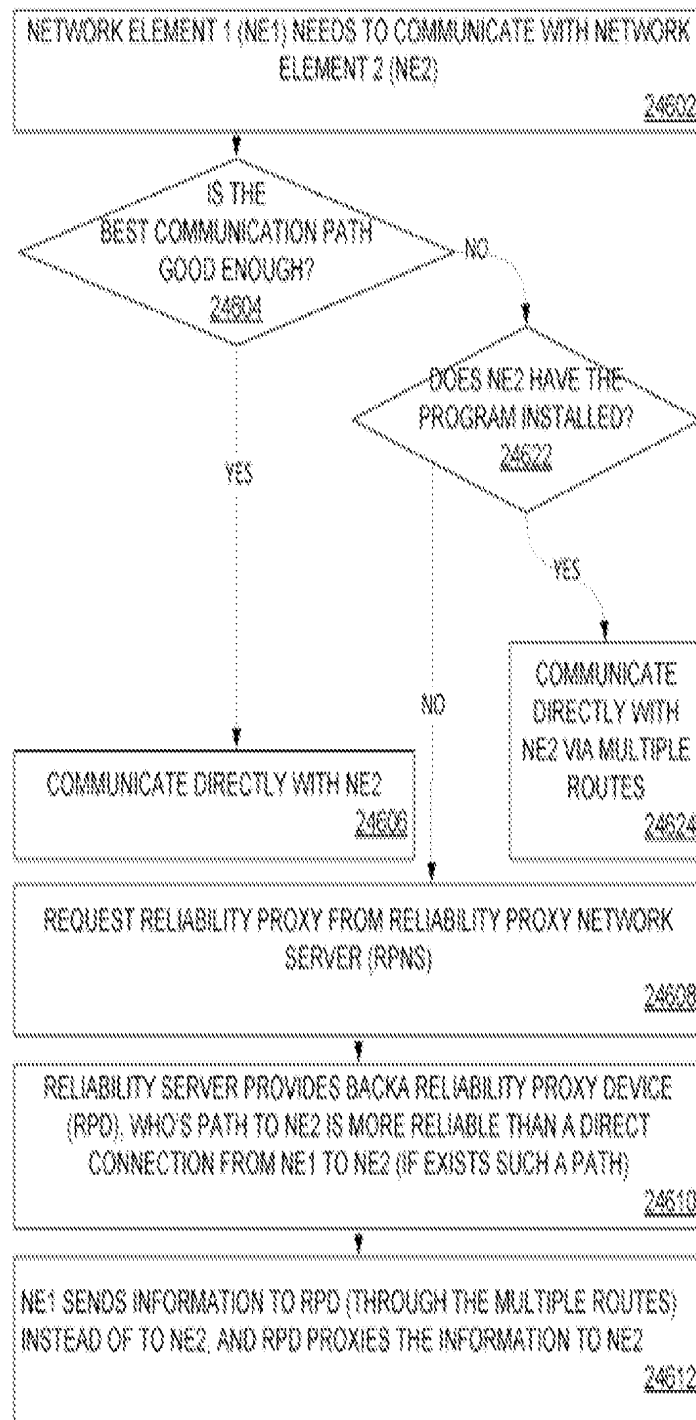


FIG. 88

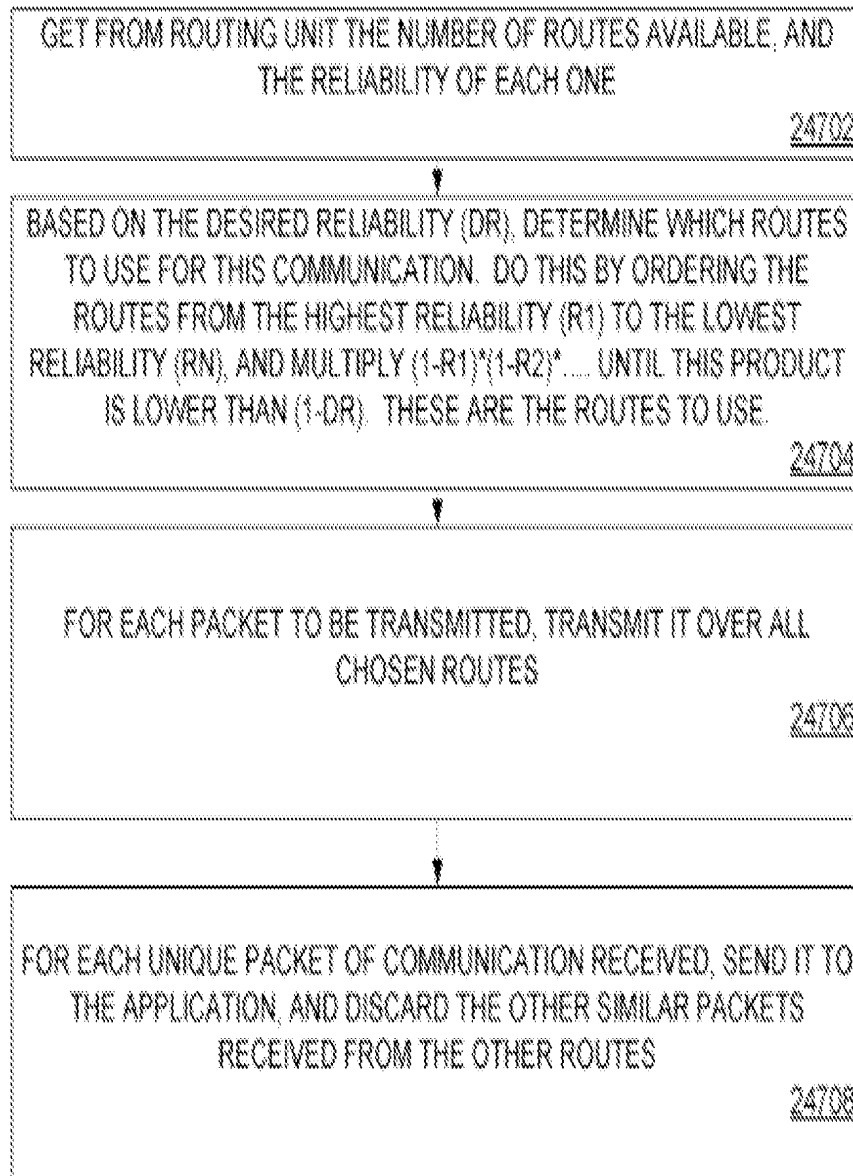


FIG. 89

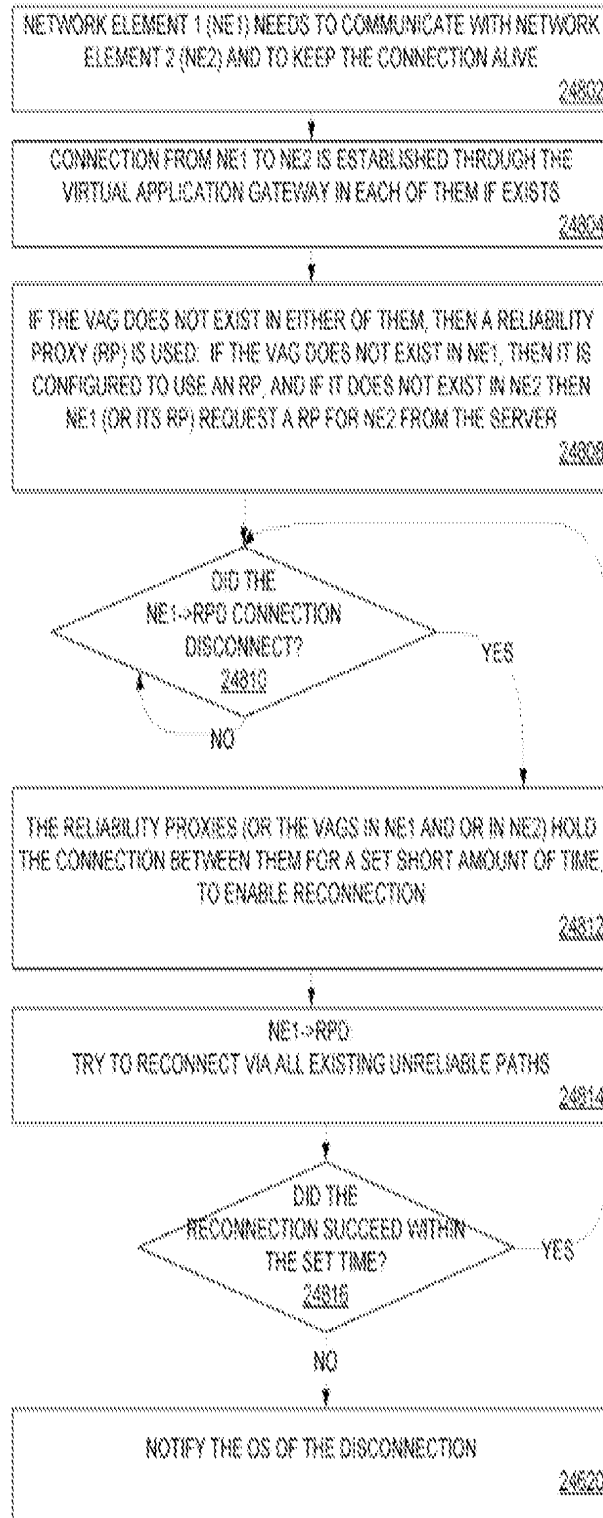


FIG. 90

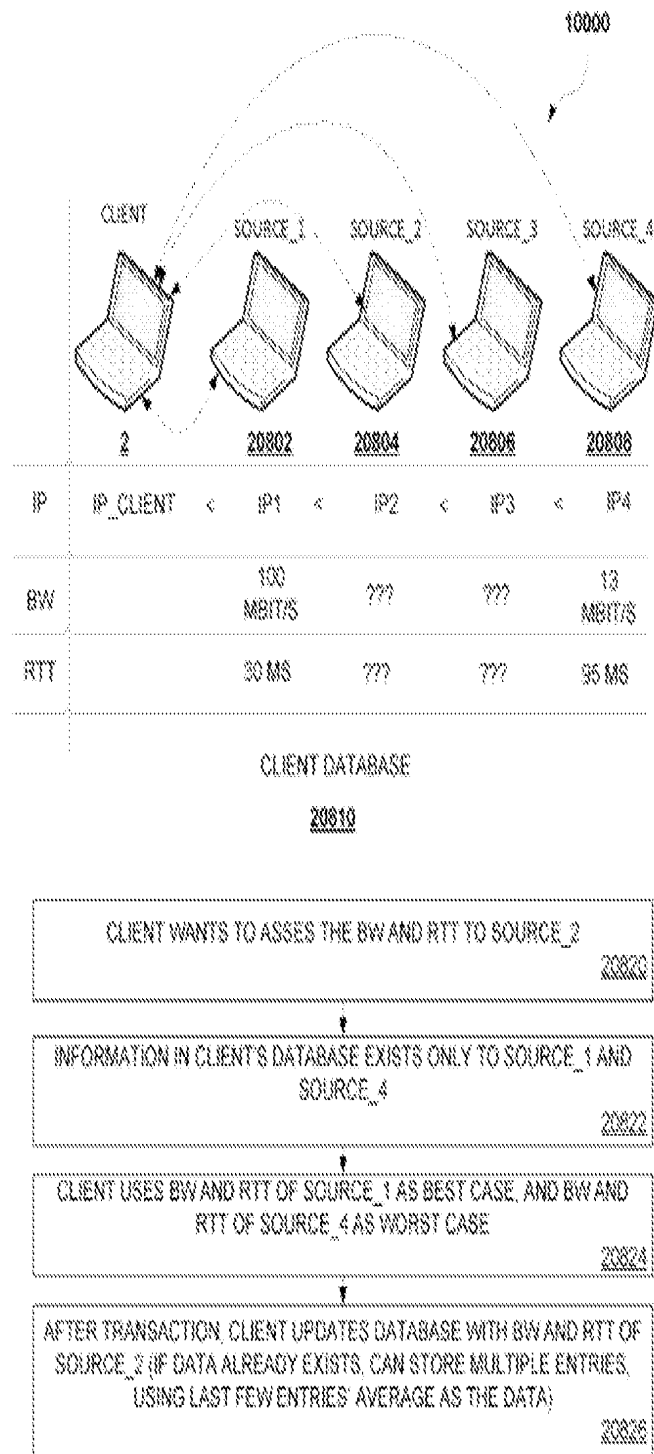


FIG. 91

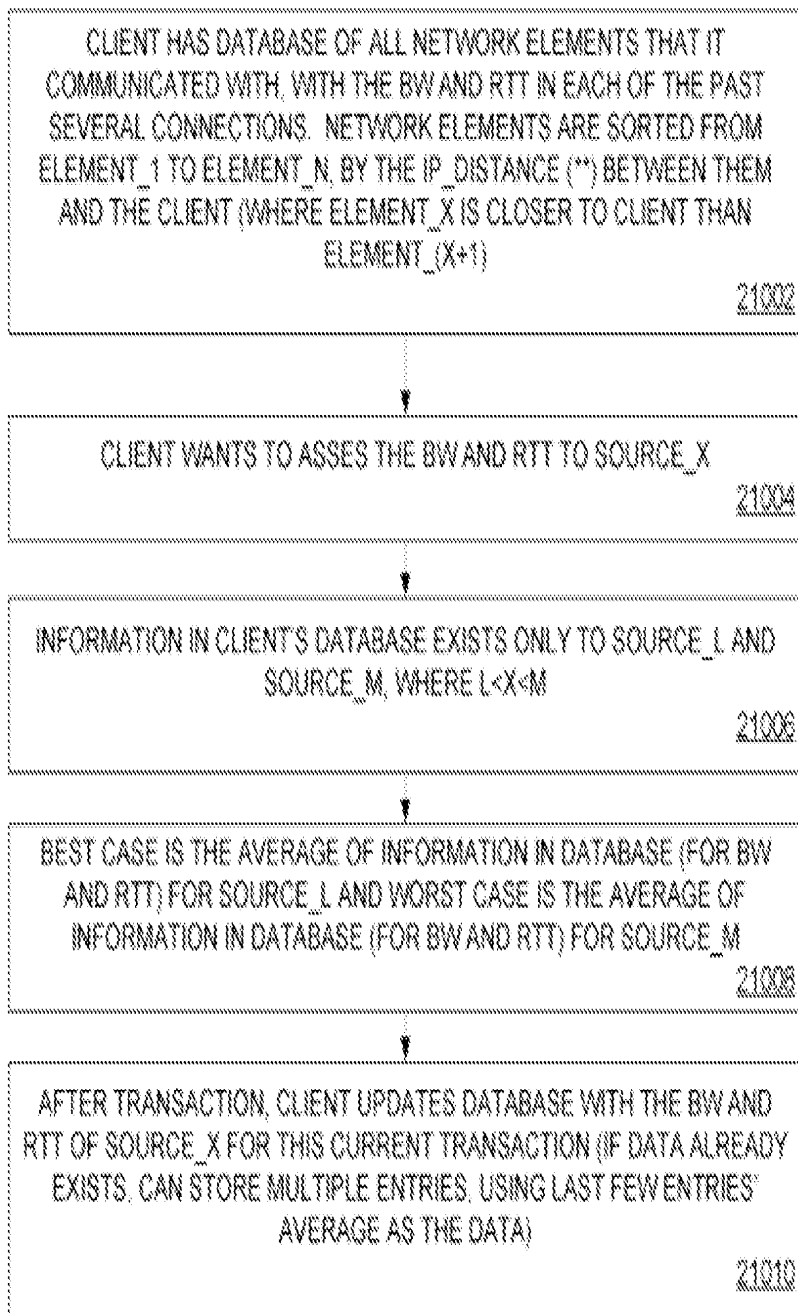


FIG. 92

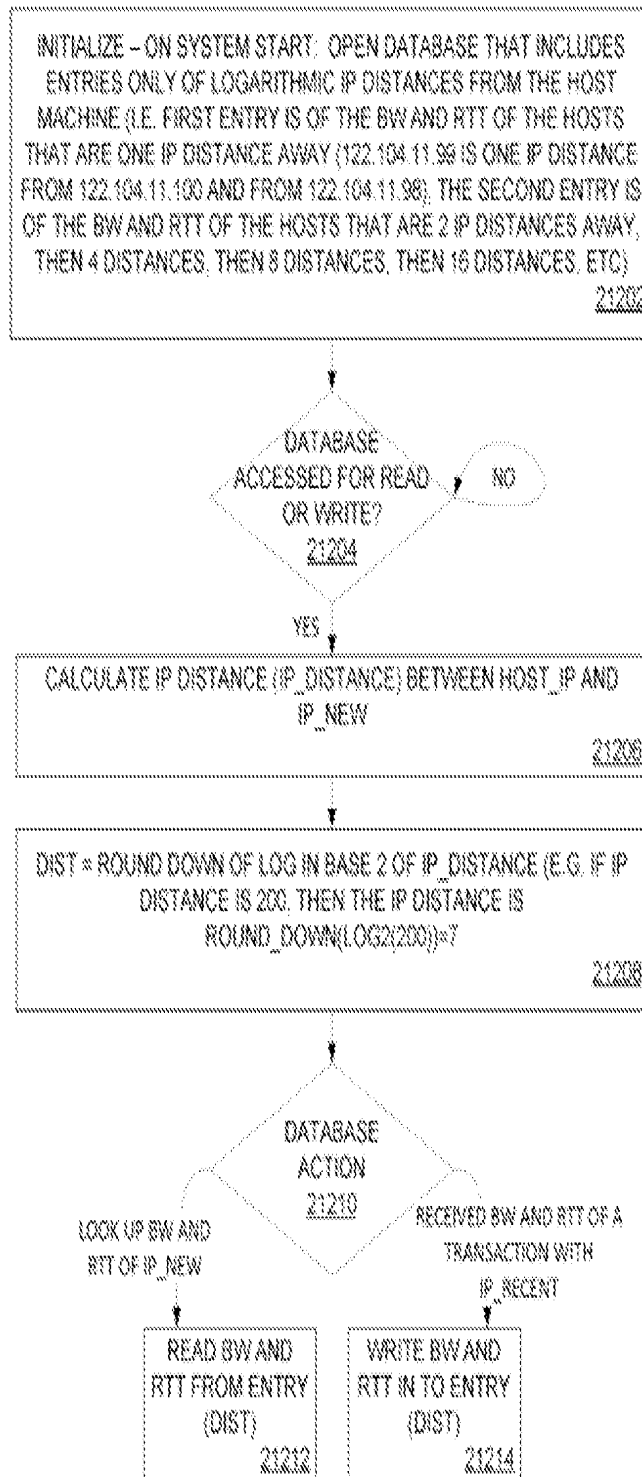


FIG. 93

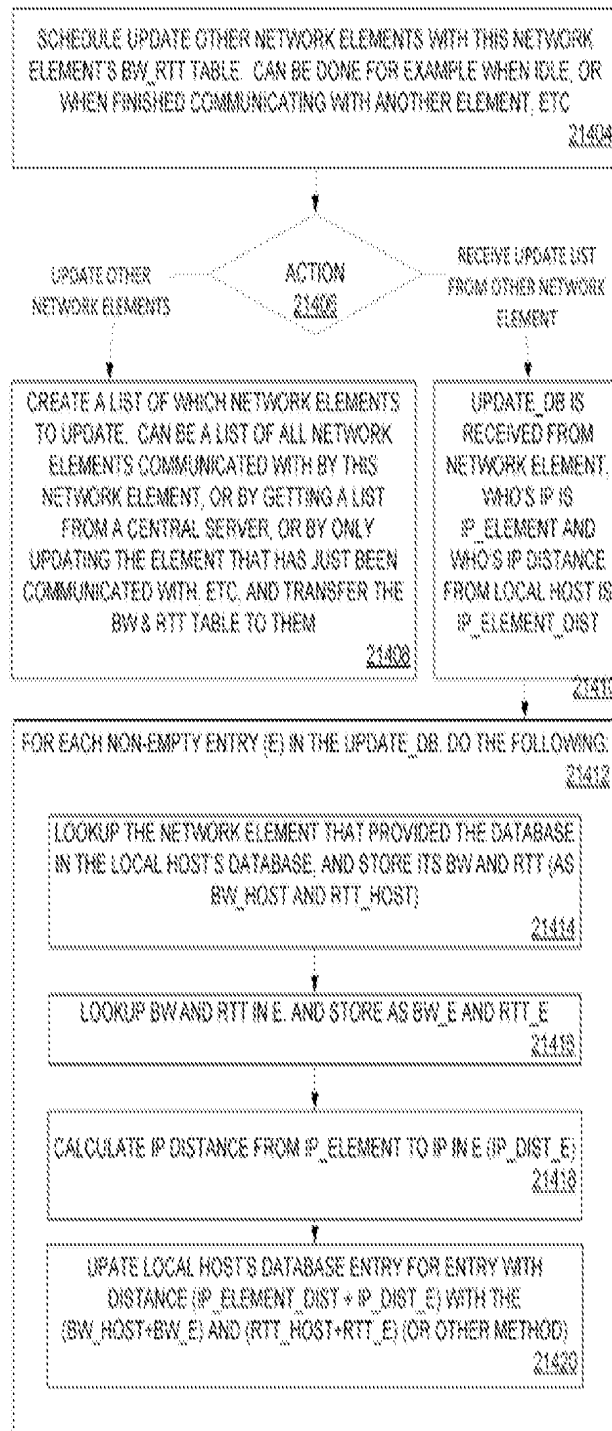


FIG. 94