

#### FIG. 55 – ACCELERATION MODULE

ON STARTUP:

5506

ESTABLISH CONNECTION WITH ACCELERATION SERVER AND GET TUNNEL\_LIST FOR THIS CLIENT

<u>5508</u>

ESTABLISH CONNECTION TO EACH OF THE TUNNELS

<u>5510</u>



ON RECEIVING A NEW MESSAGE FROM ANOTHER COMMUNICATION DEVICE ON THE NETWORK

<u>5512</u>

ACCORDING TO THE MESSAGE TYPE OR OTHER TAG, FIGURE OUT IF THE MESSAGE IS INTENDED FOR THIS DEVICE TO FUNCTION AS A CLIENT, AGENT, PEER, OR TUNNEL

<u>5514</u>

ACTIVATE THIS COMMUNICATION DEVICE AS AN AGENT, PEER OR TUNNEL ACCORDINGLY, AND PASS THIS MESSAGE TO IT

<u>5516</u>



ON INTERCEPTING A DATA REQUEST BEING SENT FROM APPLICATION TO DATA SERVER

<u>5518</u>

ACTIVATE THE CLIENT ALGORITHM, PASSING IT THE INTERCEPTED MESSAGE

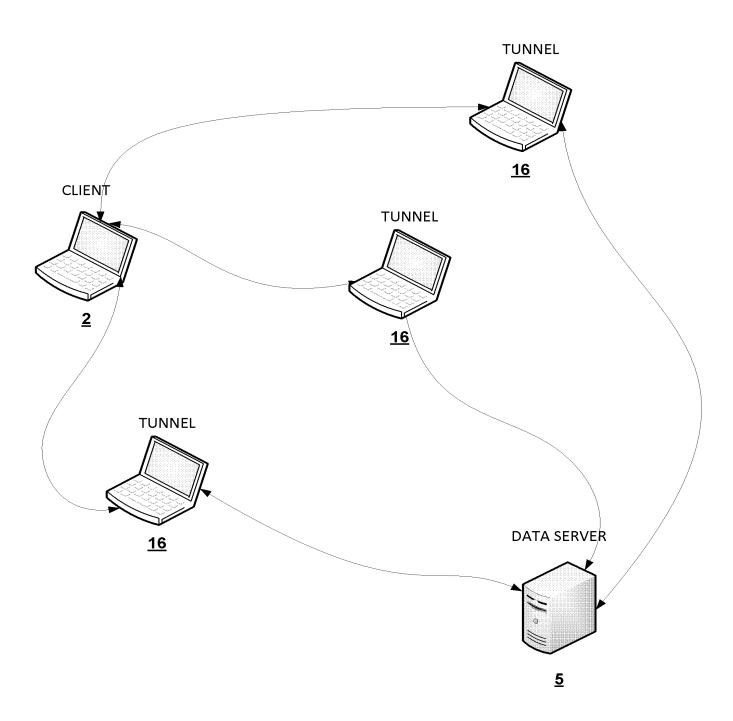
<u>5520</u>



### FIG. 56 – DATA\_REQUEST STRUCTURE

Entry Fields	Description
DATA_SERVER	THE SERVER THAT IS BEING QUERIED
REQUEST	THE REQUEST POSTED TO THE DATA_SERVER
REQUEST_META_DATA	ADDITIONAL INFORMATION FOR THE REQUEST. FOR
	EXAMPLE, FOR HTTP THIS WILL INCLUDE THE HTTP HEADERS
	AND VALIDITY
RESPONSE	THE RESPONSE DATA
RESPONSE_META_DATA	ADDITIONAL INFORMATION FOR THE REQUEST. FOR
	EXAMPLE, FOR HTTP THIS WILL INCLUDE THE HTTP HEADERS
	AND VALIDITY
CHECKSUM	THE CHECKSUM OF THE DATA OF THE RESPONSE TO THIS
	DATA_REQUEST
MAP_LIST	
	A LIST OF SEGMENTS OF DATA REPRESENTING THE FULL
	RESPONSE, WHERE FOR EACH SEGMENT, THERE IS A LIST OF
	PEERS THAT THIS CLIENT KNOWS ABOUT THAT HAVE THAT
	SEGMENT IN THEIR STORAGE. THIS INCLUDES WHETHER
	THIS CLIENT DEVICE HAS THESE SEGMENTS STORED
PEER_LIST	LIST OF COMMUNICATION DEVICES THAT MAY BE USED AS
	PEERS FOR THIS CLIENT. EACH OF THE PEERS IN THE LIST
	HAS DATA ASSOCIATED WITH IT THAT TELLS THE CLIENT
	HOW TO COMMUNICATE WITH THAT PEER, AND OTHER
	META INFORMATION ABOUT IT
TUNNEL_LIST	LIST OF COMMUNICATION DEVICES THAT MAY BE USED TO
	ASSIST IN LOADING THIS DATA_REQUEST FROM THE
	DATA_SERVER
AGENT_LIST	LIST OF COMMUNICATION DEVICES THAT MAY BE USED AS
	AGENTS FOR THIS REQUEST

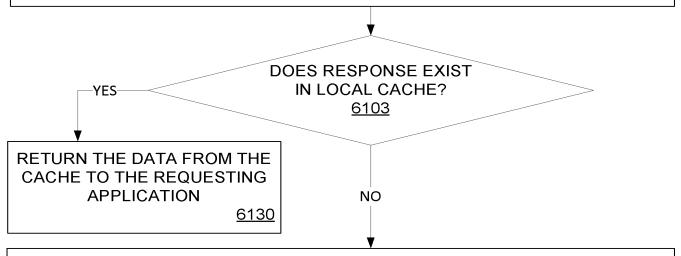
### FIG. 58 – USE OF TUNNELS



## FIG. 61 – LOADING DATA THROUGH PEERS OR THROUGH TUNNELS - FLOWCHART

THE SOFTWARE INTERCEPTS A REQUEST FOR DATA SENT BY A COMMUNICATION APPLICATION TO A DATA\_SERVER WHICH INCLUDES THE REQUEST, DATA\_SERVER, AND OTHER\_DATA (E.G. THE COOKIE IN AN HTTP DATA REQUEST)

6102



BASED ON THE DATA\_SERVER\_IP, LOCAL\_IP AND ESTIMATED SIZE OF PAYLOADS, EVALUATE WHETHER IT IS FASTER TO ACCESS THE DATA\_SERVER DIRECTLY (DIRECT), OR TO USE THE ACCELERATION NETWORK (ACCEL). IF CANNOT EVALUATE, THEN USE BOTH (EQUAL). (THIS CAN BE DONE WITH THE SPEED STATS ENGINE FOR EXAMPLE)

6104

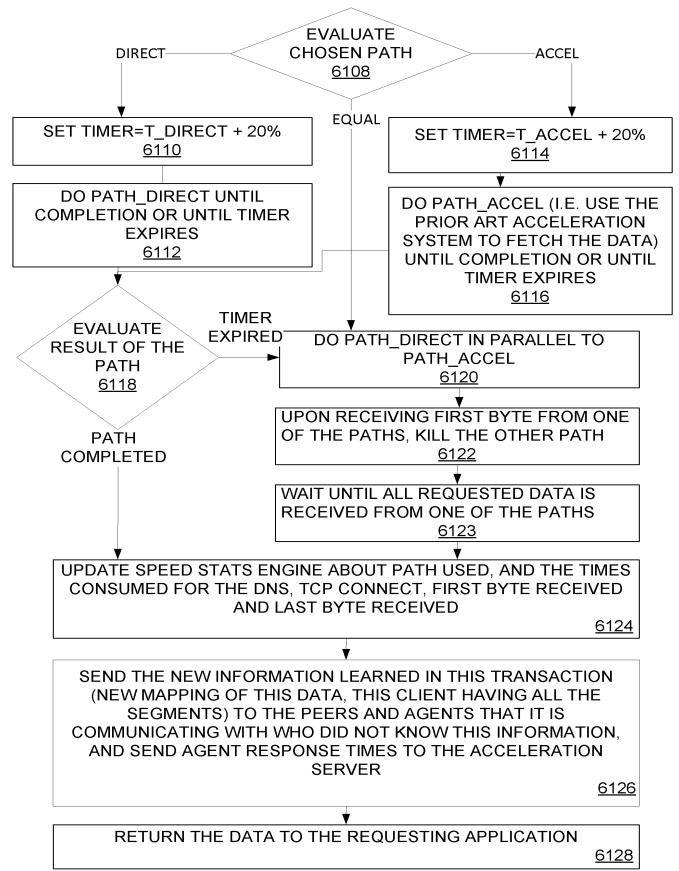
SET T\_DIRECT & T\_ACCEL TO THE EXPECTED TIME TO RECEIVE DATA VIA BOTH ROUTES

٧

<u>6106</u>

CONTINUE TO BLOCK 6108 IN FIGURE 62

### FIG. 62 – TUNNELS VS. PEERS FLOWCHART (CONT.)



## FIG. 63 – ALGORITHMS FOR SELECTING THE TUNNELS

#### **ALTERNATIVE**

6302

ACCELERATION SERVER RECEIVES REQUEST FOR TUNNELS

6304

RETURN "LOCALHOST" TO CLIENT, MEANING THAT IT IS THE ONLY TUNNEL FOR ITSELF

6306

#### SECONDARY ALTERNATIVE

6308

ACCELERATION SERVER RECEIVES REQUEST FOR TUNNELS

<u>6310</u>

CREATE LIST OF TUNNELS (TUNNEL\_LIST) BY CHOOSING THE 5
CLOSEST IPS IN THE ACCELERATION SERVER DATABASE WHICH WILL
BEST SERVE THE CLIENT (FOR EXAMPLE BY USING THE SPEED STATS
ENGINE), AND THATARE CURRENTLY ONLINE

6312

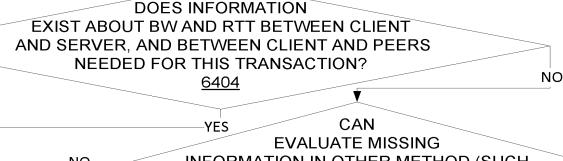
RETURN TUNNEL LIST

#### FIG. 64 – SPEED STATS ENGINE

RECEIVE CLIENT REQUEST, INCLUDING THE WEBSERVER IP, LOCAL IP AND PARTICIPATING NETWORK ELEMENTS (AGENTS, PEERS, TUNNELS) 6402

LOOK UP THE PREVIOUS INFORMATION GATHERED ABOUT THE PARTICIPATING NETWORK ELEMENTS (RESPONSE TIME, SUCCESS OF LAST TRANSACTION, ETC.)

6403



INFORMATION IN OTHER METHOD (SUCH NO-AS GEO IP)? <u>640</u>6 **RETURN "EQUAL"** 6408 YES

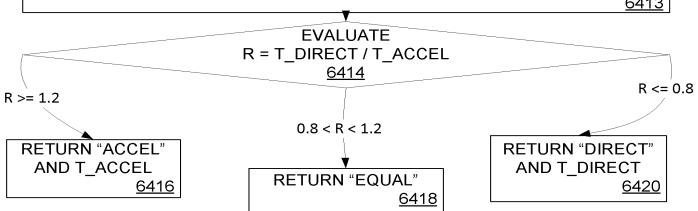
BASED ON THE INFORMATION. ESTIMATE THE TIME IT WOULD TAKE TO BRING THE PAYLOAD VIA THE TUNNELS (T DIRECT) AND VIA THE ACCELERATION NETWORK (T ACCEL)

6410

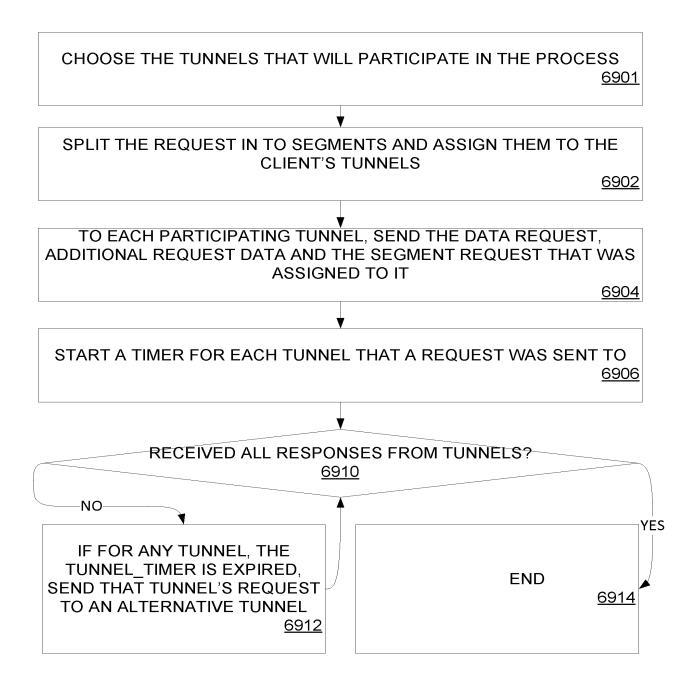
WHEN THERE ARE MORE THAN ONE DATA POINTS FOR A TIME IT WOULD TAKE FOR STAGES IN THE TRANSACTION PROCESS, USE THE AVERAGE OF THE TIME UNDER WHICH 95% OF THE SAMPLE ARE. FOR EXAMPLE, IF 95% OF THE TCP CONNECT TIMES ARE UNDER 0.23 SECONDS, THAN USE 0.23 SECONDS AS THE TIME.

6412

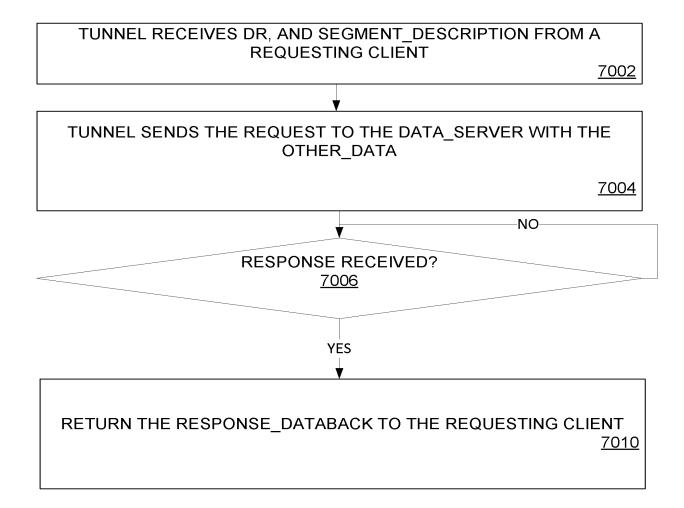
FOR THE ESTIMATION OF THE ROUND TRIP TIME, USE THE PAYLOAD UPSTREAM/BANDWIDTH UPSTREAM + PAYLOAD DOWNSTREAM/BANDWIDTH DOWNSTREAM



### FIG. 69 – PATH\_DIRECT

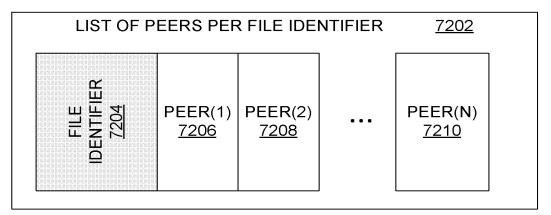


#### FIG. 70 – ACCELERATION TUNNEL



## FIG. 72 – PEER MAPS – PRIOR ART - DIAGRAMS

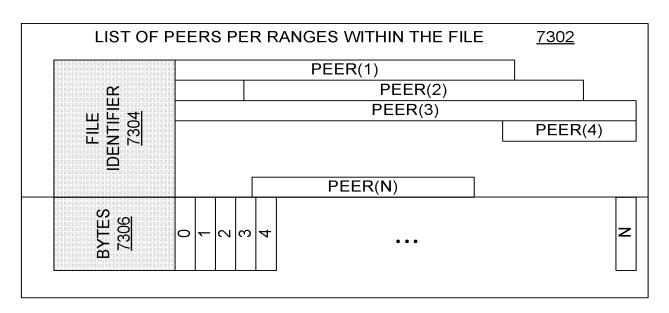
PRIOR ART – FOR EACH FILE, A LIST OF PEERS THAT HAVE CACHED ALL OF THE FILE



PROBLEM WITH PRIOR ART: ONLY PEERS THAT HAVE CACHED EVERY BYTE OF THE FILE CAN SERVE AS PEERS FOR A CLIENT REQUEST FOR THE FILE

#### FIG. 73 – PEER MAPS – DIAGRAMS

PEER MAP – FOR EACH FILE, A LIST OF PEERS THAT HAVE PARTIAL OR COMPLETE RANGES OF THE DATA FOR THAT FILE



BENEFIT: PEERS THAT HAVE CACHED EVEN PART OF THE FILE CAN SERVE AS PEERS FOR A CLIENT REQUEST FOR THIS FILE, THUS PROVIDING MORE PEERS ON AVERAGE PER REQUEST

## FIG. 74 – PEER MAPS - FLOWCHART

WHEN CLIENT HAS A PEER MAP, FLOWCHART FOR HOW TO GET THE INFORMATION FROM THE PEERS

SPLIT THE FILE IN TO RANGES, SUCH THAT THERE IS ONE PEER PER RANGE. FOR RANGES WITHOUT PEERS, MARK THEM AS "OTHER"

7402

SEND EACH RANGE REQUEST TO THE PEER ALLOCATED TO IT

7404

IF PEER DOES NOT PROVIDE THE DATA FOR THE RANGE REQUESTED, REQUEST THE RANGE FROM OTHER PEERS THAT TOGETHER HAVE THE RANGE THAT WAS REQUESTED

7406

FOR PEERS MARKED AS "OTHER" AND FOR RANGES THAT WERE NOT RECEIVED (FOR EXAMPLE BECAUSE NO PEERS PROVIDED DATA FOR IT), GET IT DIRECTLY FROM THE WEB, OR IN ANOTHER MANNER (E.G. TUNNELS)

### FIG. 76 – PEER MAPS – ALLOCATING RANGES TO PEERS BY BW & RTT - FLOWCHART

#### CREATING PEER REQUESTS OR TUNNEL REQUESTS

RECEIVE PEER MAPS FOR THE FILE WHICH INCLUDES WHICH NETWORK ELEMENTS HAVE PORTIONS OF THE FILE (OR IN CASE OF FIRST TIME LOADS, A LIST OF TUNNELS)

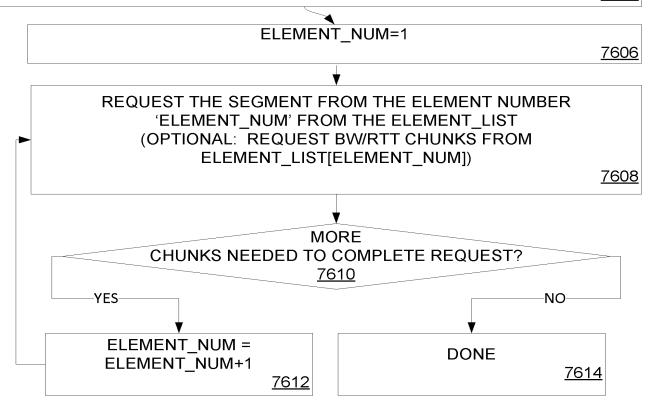
7602

FOR EACH SEGMENT, CREATE LIST OF ELEMENTS THAT HAVE CACHED THE SEGMENT (ELEMENT\_LIST), SORTED BY BW/RTT

7604

FOR EACH SUCH ELEMENT\_LIST (I.E. FOR EACH SEGMENT), DO THE FOLLOWING:

7605



NOTE: FOR LOADING INFORMATION FROM TUNNELS, IT IS BENEFICIAL TO ALSO EVALUATE THE BW/RTT FROM THE TUNNEL TO THE DATA SERVER

NetNut Ltd. v. Bright Data Ltd. V. Bright Data

Page 13 of 121

#### FIG. 78 – PEER SELECTION BY AGENT

AGENT CREATES LIST OF PEERS (PEER\_LIST) THAT HAVE THE DATA (OR PORTIONS OF THE DATA) THAT IS REQUESTED FROM THE CLIENT 7802

AGENT PRUNES THE PEER\_LIST TO X PEERS

7804

AGENT SENDS THE PRUNED PEER\_LIST TO THE CLIENT

7806

CLIENT USES THE PEER\_LIST TO FETCH THE DATA

7808

METHODS OF DETERMINING WHICH PEERS ARE IN THE PEER\_LIST: 7810

THE 'X' PEERS THAT ARE CLOSEST TO THE CLIENT

7812

THE 'X' PEERS THAT HAVE THE FASTEST CONNECTION TO THE CLIENT (COULD BE SORTED BY BW/RTT FOR EXAMPLE)

**7814** 

THE 'X' PEERS THAT MOST RECENTLY PROVIDED THIS DATA

7816

METHODS OF DETERMINING 'X' - THE NUMBER OF PEERS IN THE PEER\_LIST:

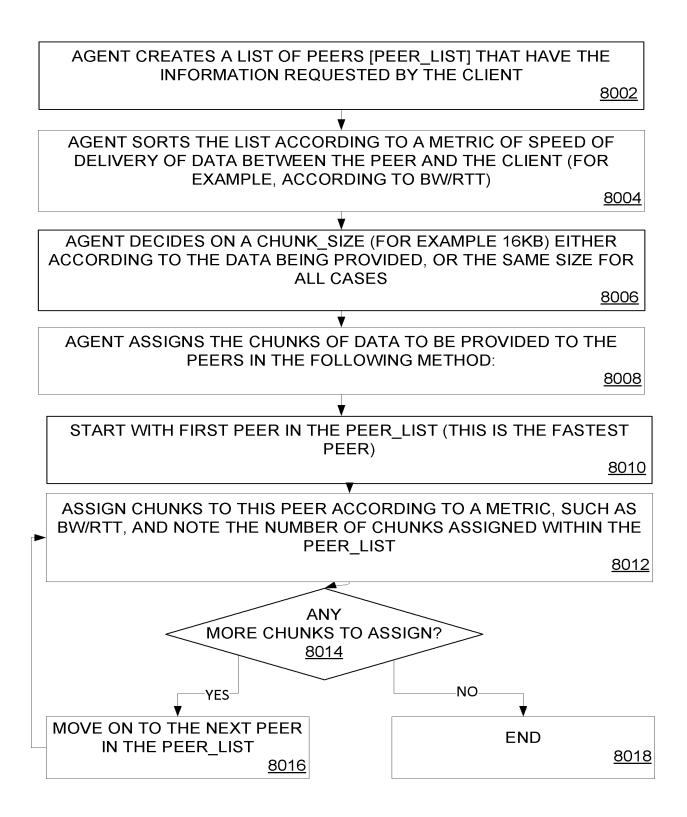
<u> 7818</u>

ALWAYS AS 5 PEERS, AND CLIENT ASKS FOR MORE IF NEEDED

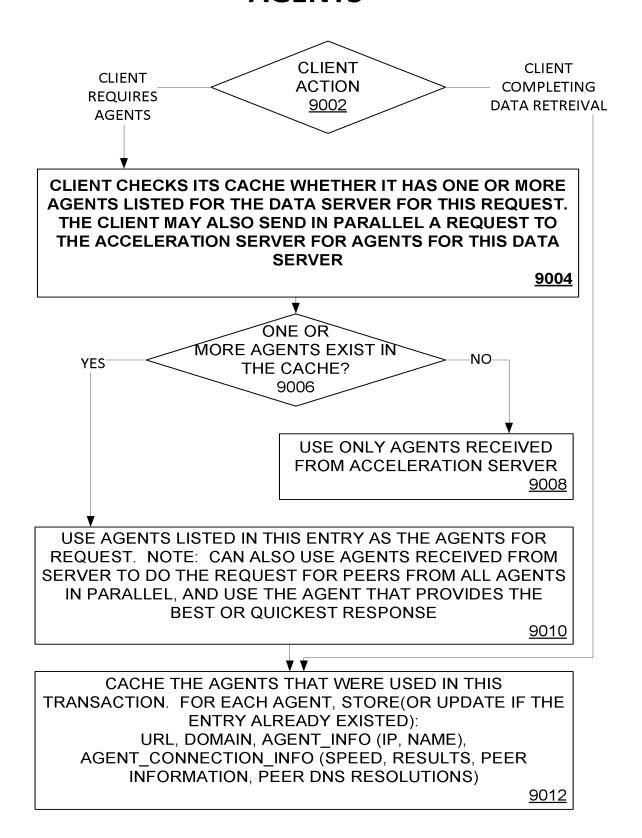
7820

ACCORDING TO THE FILE SIZE, ALLOCATING A PEER FOR EVERY S BYTES (S CAN BE 100KB FOR EXAMPLE)

## FIG. 80 – PEER ALLOCATION ACCORDING TO BW/RTT



### FIG. 90 – CACHING PREVIOUSLY USED AGENTS

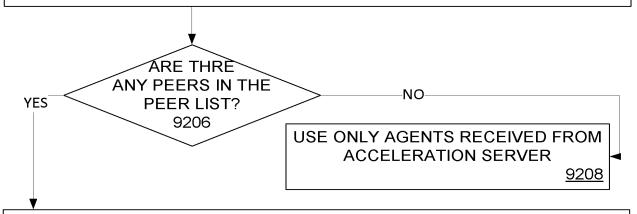


## FIG. 92 – PASSIVE AGENTS – USING KNOWLEDGEABLE PEERS AS AGENTS



IN PARALLEL TO SENDING REQUEST TO ACCELERATION SERVER,
CLIENT CHECKS IN ITS CACHE TO CREATE A LIST OF ALL PEERS IT HAS
COMMUNICATED TO, AND CREATES A LIST OF THESE PEERS
[PEER\_LIST]. THIS LIST MAY BE REDUCED TO ONLY THE PEERS THAT
HAVE PROVIDED INFORMATION ABOUT THE DATA SERVER OF THE
CURRENT REQUEST

<u>9204</u>



USE PEERS LISTED IN THIS ENTRY AS THE AGENTS FOR REQUEST (E.G. PEERS THAT PROVIDED INFORMATION FROM THIS DOMAIN, WILL ALSO BE AGENTS FOR IT, SINCE THEY MAY HAVE THE DATA).

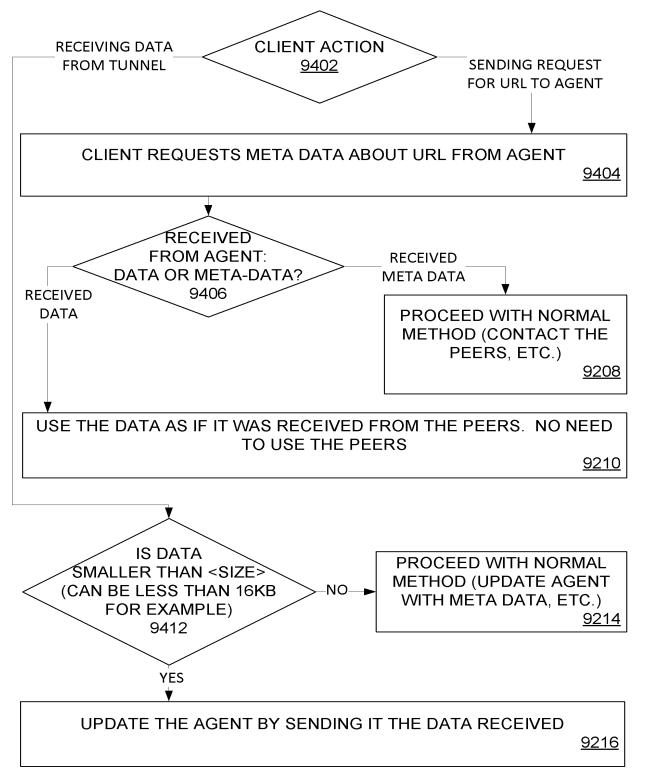
NOTE 1: CAN ALSO USE AGENTS RECEIVED FROM SERVER TO DO THE REQUEST FOR PEERS FROM ALL AGENTS IN PARALLEL, AND USE THE AGENT THAT PROVIDES THE BEST OR QUICKEST RESPONSE

9210

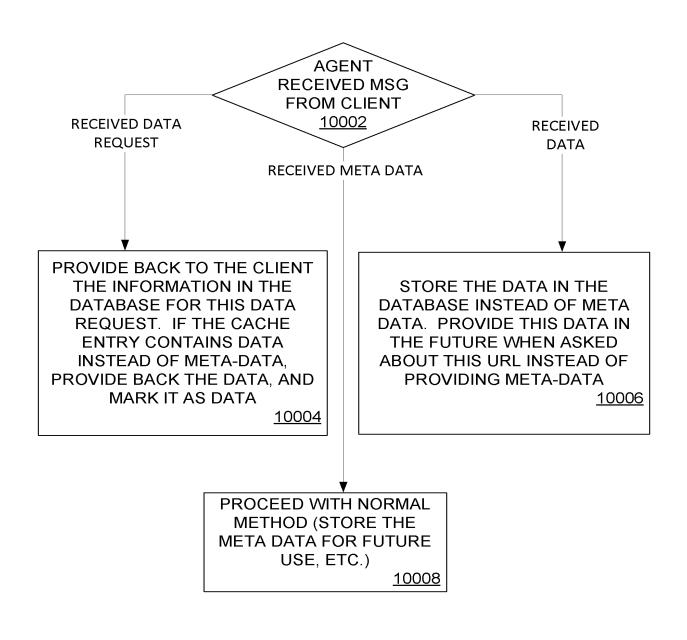
CACHE THE PEERS THAT WERE USED IN THIS TRANSACTION INTO 'PEERS DATABASE' (PEERS\_DB). FOR EACH PEER, STORE(OR UPDATE IF THE ENTRY ALREADY EXISTED):

URL SERVED, DOMAIN SERVED, AGENT\_INFO (IP, NAME),
AGENT\_CONNECTION\_INFO (SPEED, RESULTS, PEER INFORMATION,
PEER DNS RESOLUTIONS), PEER CONNECTION INFO (SPEED, RESULTS,
PEER INFORMATION, PEER DNS RESOLUTIONS)

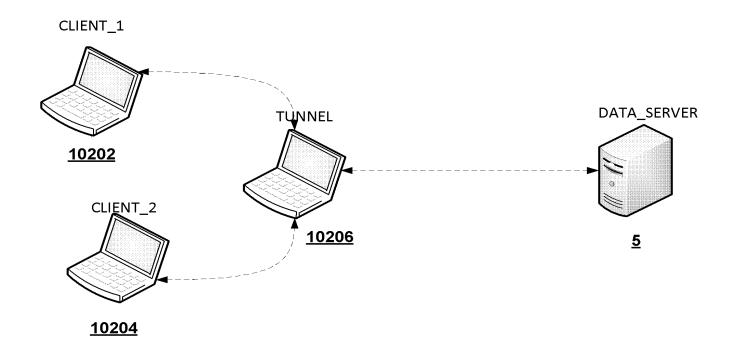
#### FIG. 94 – INLINE DATA – CLIENT SIDE



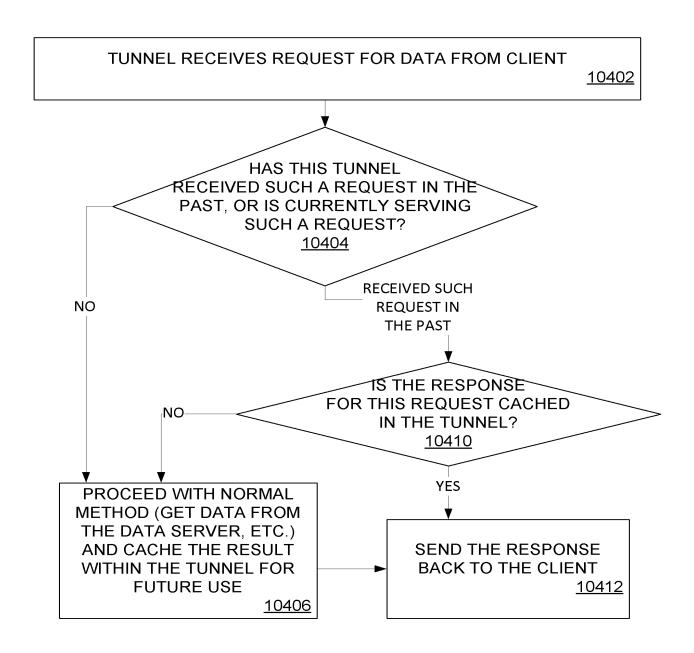
#### FIG. 100 – INLINE DATA – AGENT SIDE



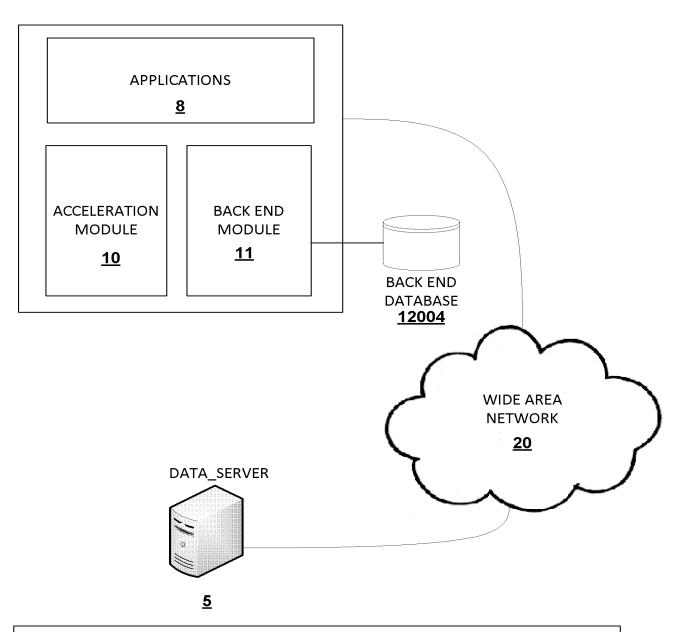
## FIG. 102 – TUNNEL CACHING – NETWORK SETUP



## FIG. 104 – TUNNEL CACHING – FLOWCHART



# FIG. 120 – USING BACK ENDS TO CUSTOMIZE SPECIFIC DATA REQUESTS - DIAGRAM

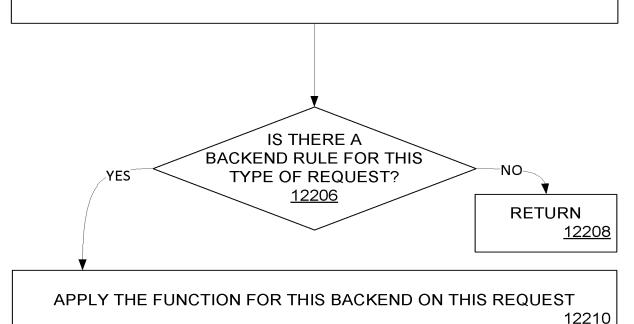


#### **CUSTOMIZE BY:**

- URL
- DOMAIN
- DATA SERVER IP
- TYPE OF FILE (.JPG, .FLV)
- TIME OF DAY
- GEOGRAPHY OF CLIENT OR SERVER OR BOTH

# FIG. 122 – USING BACK ENDS TO CUSTOMIZE SPECIFIC DATA REQUESTS - FLOWCHART

EVALUATE THE REQUEST FROM APPLICATION TO THE ACCELERATION MODULE BASED ON THE PARAMETERS OF THE REQUEST (URL, DOMAIN, DATA SERVER IP, TYPE OF FILE, TIME OF DAY, GEOGRAPHY OF CLIENT AND SERVER, ETC.), CHECK IN THE BACKEND DATABASE WHETHER THERE IS A BACKEND RULE FOR THIS TYPE OF REQUEST 12202



## FIG. 124 – SPECIFIC BACK END – USAGE STATS BACKEND

A REQUEST FOR DATA WHICH UPDATES THE USAGE STATS OF THE SITE, SUCH AS GOOGLE ANALYTICS IS IDENTIFIED

<u>12402</u>

LOOK UP THIS DATA REQUEST IN THE STATS\_BACKEND DATABASE,
AND RETREIVE THE 'TYPICAL\_RESPONSE'

<u>12404</u>

RETURN TO THE REQUESTING APPLICATION THE TYPICAL\_RESPONSE

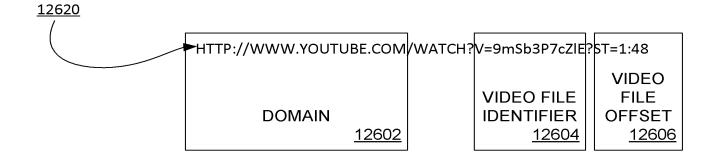
12406

SEND THE REQUEST TO THE STATS\_SERVER (IMPROVED IMPLEMENTATION IS TO DO THIS WHEN THE NETWORK IS IDLE, SINCE THIS IS NON-CRITICAL INFORMATION)

12408

WAIT FOR REPLY FROM STATS\_SERVER, AND DO NOTHING WITH IT WHEN IT RETURNS

## FIG. 126 – SPECIFIC BACK END – VIDEO OFFSET BACKEND – PRIOR ART



PROBLEM #1: IF A VIDEO FILE IS REQUESTED WITH AN OFFSET, IT CANNOT BE CACHED AS A URL SINCE EVERY DIFFERENT OFFSET IS A DIFFERENT URL.

12608

PROBLEM #2: OFFSETS CAN BE BY TIME OFFSET, BYTE OFFSET OR RELATIVE OFFSET. FOR CACHING AND RETREIVAL, HOW IS THE NORMALIZATION BETWEEN THESE DONE?

<u>12610</u>

## FIG. 128 – SPECIFIC BACK END – VIDEO OFFSET BACKEND – CACHING BY URL

LOOK UP THE DOMAIN OF THE URL IN THE VIDEO BACKEND DATABASE, AND IDENTIFY THE ADDITIONAL BREAKDOWN OF THE URL 12802

S = DOMAIN + "/" + VIDEO\_FILE\_IDENTIFIER
(E.G. HTTP://WWW.YOUTUBE.COM/WATCH?V=9MSB3P7CZIE?ST=1:48
BECOMES WWW.YOUTUBE.COM/9MSB3P7CZIE)

12804

(OPTIONAL) CACHE THE DATA WITH S AS THE URL. ALSO CACHE THE STARTING AND ENDING OFFSET FOR THE DATA THAT IS CACHED 12806

### FIG. 130 – SPECIFIC BACK END – VIDEO OFFSET BACKEND – NORMALIZING OFFSET

TIME OFFSET:

#T=3M54S

B=10344

BYTE OFFSET: 13004

D-10044

RELATIVE OFFSET:

R = 54.3

13006

13002

URL WITH OFFSET IS REQUESTED OR RECEIVED (E.G. HTTP://WWW.YOUTUBE.COM/WATCH?V=9MSB3P7CZIE?ST=1:48)

<u>13008</u>

**V** 

NORMALIZE TO URL TO NEW\_URL ACCORDING TO THE ABOVE 'CACHING BY URL' ALGORITHM

(E.G. HTTP://WWW.YOUTUBE.COM/WATCH?V=9MSB3P7CZIE?ST=1:48 BECOMES WWW.YOUTUBE.COM/9MSB3P7CZIE)

13010

 $\overline{\downarrow}$ 

SEND REQUEST (NEW\_URL) TO DATA SERVER, AND IMMEDIATELY CLOSE IT,
TO GET FULL FILE INFO, INCLUDING CONTENT LENGTH
(LENGTH\_SOURCE\_UNITS) IN THE UNITS OF THE OFFSET RECEIVED AND THE
CONTENT LENGTH (LENGTH\_DEST\_UNITS) IN THE UNITS OF THE OFFSET
DESIRED

13012

1

RELATIVE\_OFFSET = TIME\_OFFSET / TOTAL TIME
OR BYTE\_OFFSET / TOTAL BYTES
OR RELATIVE OFFSET RECEIVED

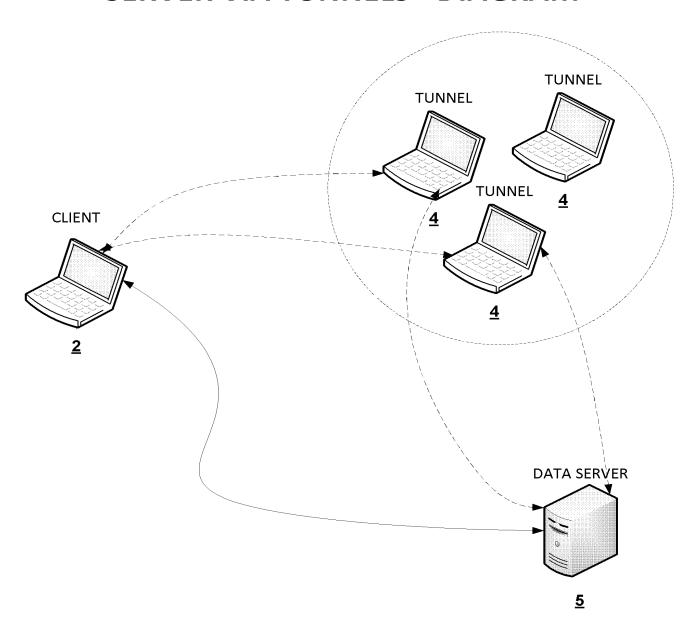
13014

NEW\_ OFFSET = RELATIVE\_OFFSET \* CONTENT\_LENGTH

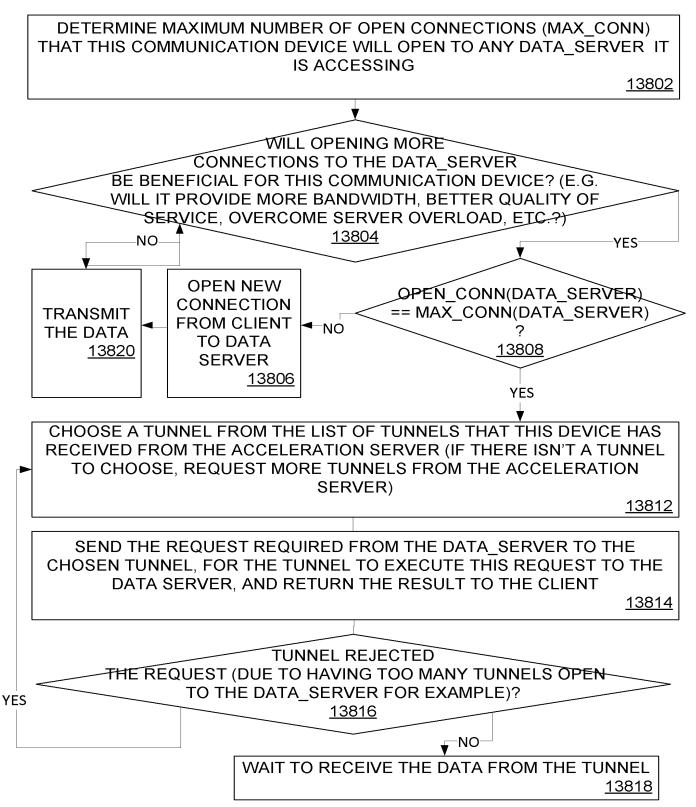
<u>13016</u>

USE NEW\_URL AND NEW\_OFFSET FOR CACHING OR RETREIVING FILE DATA FROM THE CACHE (IF CACHING THE DATA, CACHE THE DATA ONLY, WITHOUT THE HEADERS)

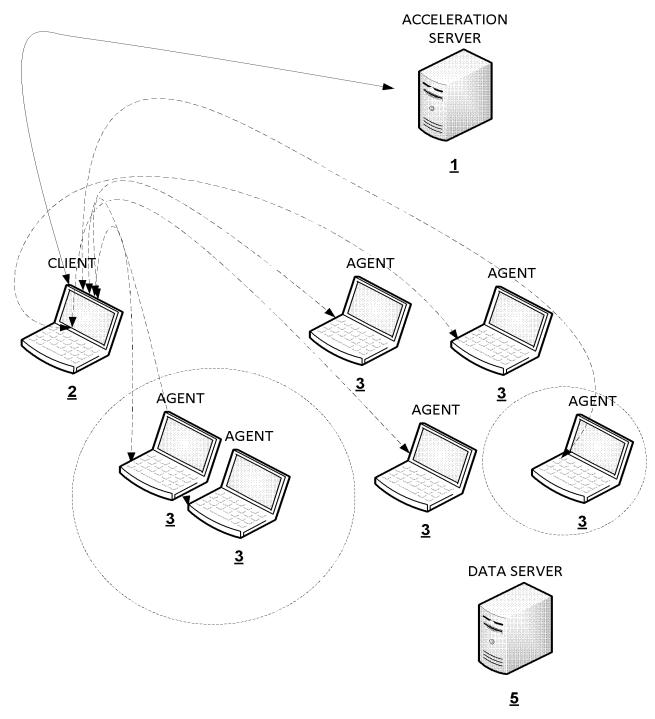
# FIG. 136 – INCREASING MAXIMUM NUMBER OF CONNECTIONS TO DATA SERVER VIA TUNNELS - DIAGRAM



# FIG. 138 – INCREASING MAXIMUM NUMBER OF CONNECTIONS TO DATA SERVER VIA TUNNELS - FLOWCHART



# FIG. 142 – AGENT SELECTION – FEEDBACK TO SERVER ABOUT AGENTS - DIAGRAM



# FIG. 144 – AGENT SELECTION – FEEDBACK TO SERVER ABOUT AGENTS – FLOWCHART FOR SERVER SIDE

ACCELERATION SERVER (AC) RECEIVES REQUEST FROM CLIENT TO PROVIDE AGENTS FOR FURTHER PROCESSING A REQUEST

14402

AC CHOOSES THE X MOST PREFERABLE AGENTS NEAR THE CLIENT. WITHIN THIS RANGE, THE AC CHOOSES X NEXT-BEST AGENTS

14404

ACCELERATION SERVER PROVIDES AGENT WITH FULL CONNECTION INFORMATION FOR THESE AGENTS

<u>14406</u>

YES

RECEIVED
RESPONSE FROM CLIENT?
14406

CLIENT RESPONSE WILL INCLUDE ALL CONNECTION INFORMATION FOR ALL THESE AGENTS (ONLINE/OFFLINE, RESPONSE FOR EACH STAGE IN THE CONNECTION, INFORMATION ABOUT DATA RECEIVED, ETC.)

<u>14408</u>

AC CACHES THIS INFORMATION ABOUT THE AGENTS FOR USE FOR FUTURE RECOMMENDATIONS TO CLIENTS

14410

CLIENTS MAY ALSO PROVIDE BACK INFORMATION ABOUT PEERS USED, FOR THEIR CONNECTION INFORMATION AS AGENTS FOR OTHER REQUESTS

# FIG. 146 – AGENT SELECTION – FEEDBACK TO SERVER ABOUT AGENTS – FLOWCHART FOR CLIENT SIDE

CLIENT SENDS REQUEST TO ACCELERATION SERVER (AC) TO RECEIVE AGENTS LIST

14602

CLIENT RECEIVES BACK 2X AGENTS, AND TRIES TO CONNECT TO ALL OF THEM

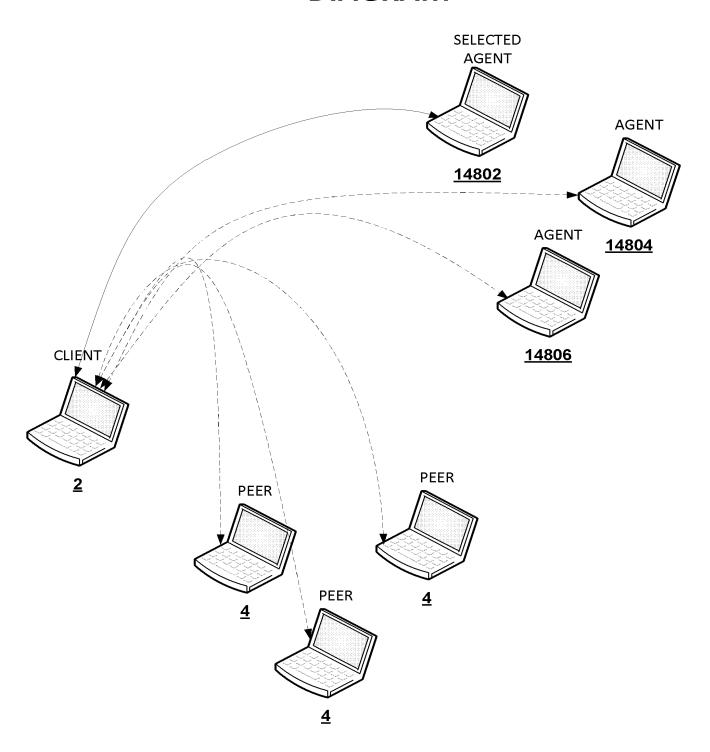
14604

CLIENT COMPLETES THE TRANSACTION AS PER HOLA'S PATENT PPP

14606

WHEN THE NETWORK IS NOT BUSY, PROVIDE BACK INFORMATION TO SERVER ABOUT CONNECTION INFORMATION OF EACH OF THE AGENTS THAT WERE IN THE LIST BY THE AC 14608

# FIG. 148 – PEER SELECTION – FEEDBACK TO AGENT ABOUT PEERS DIAGRAM



### FIG. 152 – AGENT SELECTION – FEEDBACK TO AGENT ABOUT PEERS – FLOWCHART FOR CLIENT SIDE

**CLIENT SIDE** 

CLIENT SENDS REQUEST TO AGENT TO RECEIVE PEER LIST

<u>15202</u>

CLIENT RECEIVES BACK PEERS LIST, AND USES THEM

15204

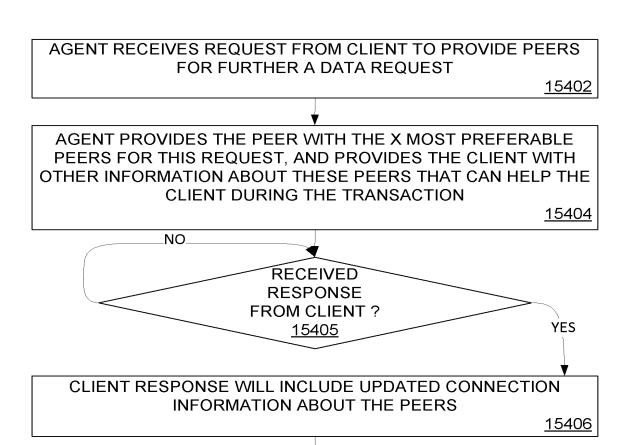
CLIENT LOGS THE INFORMATION LEARNED ABOUT EACH PEER (SUCH AS ITS IP, PORTS USED, SECURITY INFORMATION SPEED FOR VARIOUS CONNECTION PHASES, ETC.), AND COMPARES THIS INFORMATION TO THE INFORATION RECEIVED FROM THE AGENT, AND MAKES A LIST OF THE DIFFERNCES BETWEEN THE INFORATION RECEIVED AND WHAT IT LEARNED THROUGH THE TRANSACTION

15206

WHEN THE NETWORK IS NOT BUSY, CLIENT PROVIDES BACK THE LIST OF DIFFERENCES TO THE AGENT

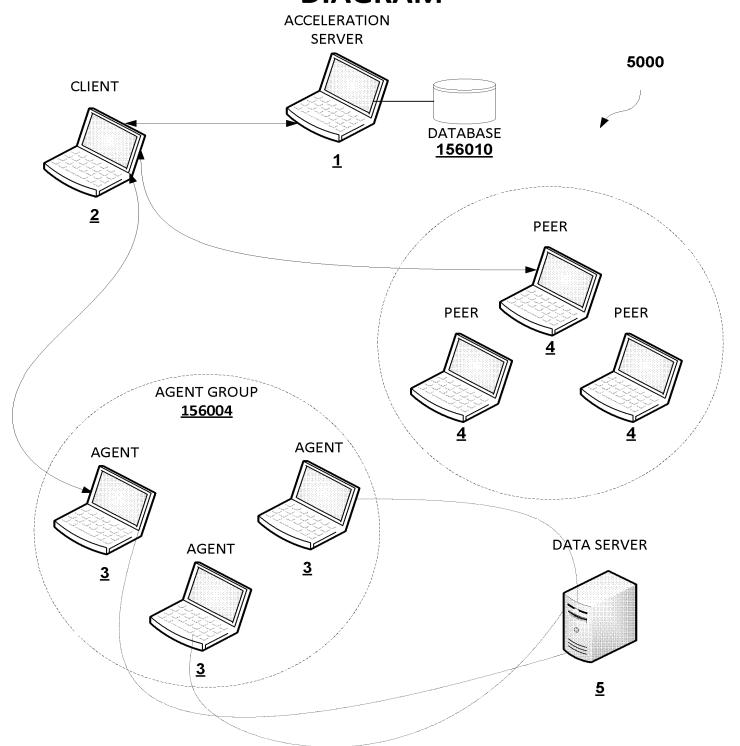
## FIG. 154 — PEER SELECTION — FEEDBACK TO AGENT ABOUT PEERS FLOWCHART

AGENT SIDE



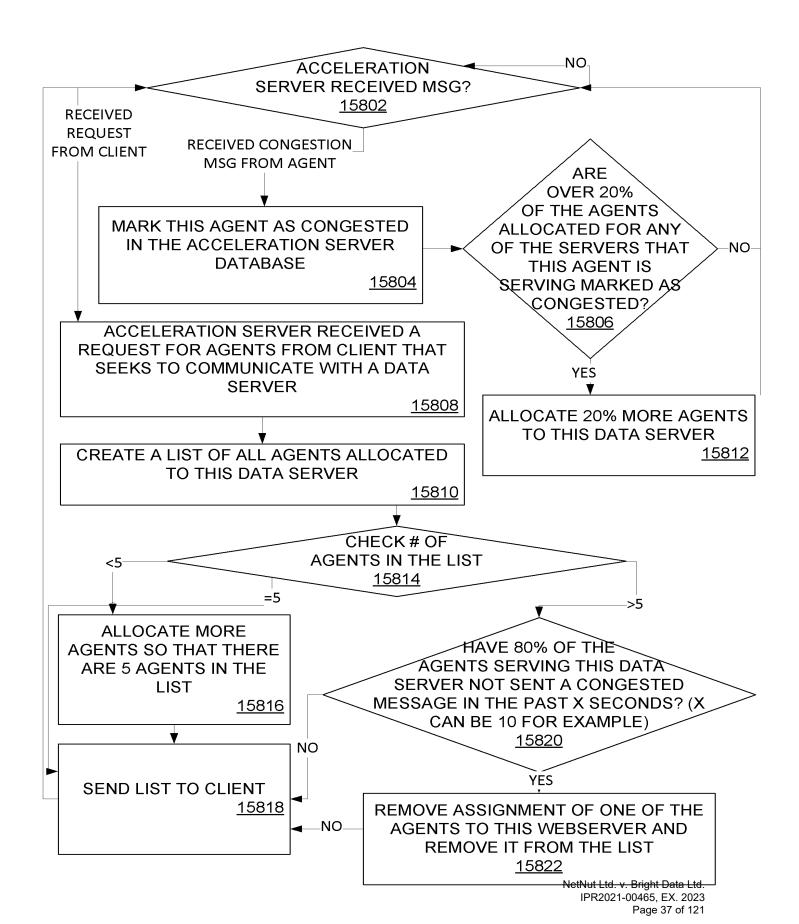
AGENT CACHES THIS INFORMATION TO PROVIDE TO OTHER NETWORK ELEMENTS THAT MAY REQUEST THIS INFORMATION, AND TO PROVIDE TO CLIENTS WHEN PROVIDING THEM WITH A PEER LIST

## FIG. 156 – LOAD BALANCING OF AGENTS - DIAGRAM



NOTE: THE AGENT GROUP IS A GROUP OF AGENTS ASSIGNED TO A SPECIFIC DATA SERVER. AS THE VARIOUS AGENTS IN THE GROUP BECOME MORE CONGESTED, MORE NETWORK ELEMENTS ARE JOINED IN TO THAT GROUP BY THE ACCELERATION SERVER. AS AGENTS FREE UP, SOME ARE RELEASED BY THE ACCELERATION SERVER AND THE GROUP BECOMES SMALLER

### FIG. 158 – LOAD BALANCING OF AGENTS - FLOWCHART



## FIG. 160 – LOAD BALANCING OF AGENTS – ADDITIONAL METHODS OF LEARNING CONGESTION

BY CLIENT

IF AGENTS IN THE LIST DO NOT RESPOND TO THE CLIENT'S REQUESTS, CLIENT NOTIFIES SERVER THAT THESE ARE CONGESTED (ACCELERATION SERVER TREATS IT AS IF THE AGENT SENT A 'CONGESTED' SIGNAL)

16002

**BY AGENT** 

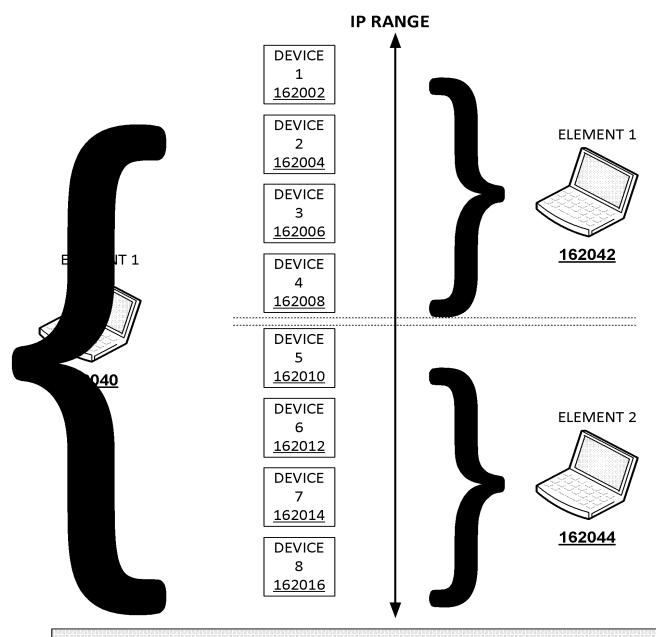
IF AGENT'S RESOURCES (E.G. STORAGE, I/O, CPU, BANDWIDTH, ETC.) ARE BEING USED FOR THE BENEFIT OF THE NETWORK (I.E. NOT FOR OWN BENEFIT) OVER A CERTAIN THRESHHOLD (E.G. ABOVE 10% OF THE CPU PROCESSING), THEN AGENT SENDS A 'CONGESTED' SIGNAL TO THE ACCELERATION SERVER

16004

BY ACCELERATION SERVER

ACCELERATION SERVER CAN PING THE AGENTS AND MARK THEM AS CONGESTED IF THEY DO NOT REPLY ABOVE A CERTAIN TIME (E.G. ABOVE 300MS)

#### FIG. 162 – AGENT LOAD REDUCTION: REDUCING NUMBER OF IPS PER AGENT -DIAGRAM



#### **NOTES:**

- 1. AGENT 1 IS BEING ACCESSED BY A RANGE OF IPS [IP(1) TO IP(8)] AND IS OVERLOADED
- 2. THERE ARE OTHER AGENTS THAT ARE FREE AND CAN SHARE THE LOAD [AGENT 2]
- 3. AGENT 1 DIVIDES THE IP RANGE IN TO 2 EQUAL RANGES [IP(1)...IP(4) AND IP(5)...IP(8)] AND ASSIGNS THE FIRST RANGE TO HIMSELF AND THE SECOND TO AGENT 2
- 4. ALL SECOND RANGE IPS WILL BE REFERRED TO AGENT2 BY THE AGENTS AND ACCELERATION SERVER, AND WILL BE REDIRECTED TO AGENT2 BY AGENT1 AND AGENT1 WILL NOTIFY THE REFERRER TO REDIRECT TO AGENT2 NEXT TIME

NetNut Ltd. v. Bright Data Ltd. 156006 IPR2021-00465, EX. 2023

Page 39 of 121

#### FIG. 164 – AGENT LOAD REDUCTION: REDUCING NUMBER OF IPS PER AGENT -FLOWCHART

#### **ELEMENT IS CONGESTED**

16402

ELEMENT SORTS ALL DEVICES THAT HAVE ACCESSED IT ACCORDING TO GEO IP, SO THAT IF X<Y THEN GEO\_IP(X)<GEO\_IP(Y) [FROM HERE ON IN THE FLOWCHART, IP REFERS TO GEO\_IP – I.E. BY GEOGRAPHIC CLOSENESS]

16404

ELEMENT CHOOSES THE DEVICE THAT IS THE "MIDDLE" OF THIS RANGE [PEER(MIDDLE)] "MIDDLE" CAN BE BY:

- MEDIAN IP (I.E. THE IP THAT IS CLOSEST TO THE MIDDLE OF THE IP RANGE BETWEEN IP(1) AND IP(N)
- MIDDLE IP (I.E. IP(N/2))
- THE IP FOR WHICH THE NUMBER OF REQUESTS BELOW AND ABOVE IT ARE THE SAME (I.E. SUM(REQ(IP(1...X))=SUM(REQ(IP(X+1...N))

16406

ELEMENT CHOOSES A DIFFERENT ELEMENT THAT IS FREE [FREE\_ELEMENT] FROM THE ELEMENT CANDIDATE LIST (I.E. A LIST OF ELEMENTS THAT IT MAKES SENSE TO ASSIGN IN THIS CASE). THIS COULD BE ONE OF:

- ANY ONE OF THE FREE ELEMENTS (I.E. OF ELEMENTS NOT CURRENTLY ASSIGNED TO ANY IP RANGE)
- A FREE ELEMENT THAT IS CLOSE TO THE DEVICES ACCESSING IT, OR TO THE DATA SERVER IT IS ACCESSING
- AN ELEMENT THAT IS NOT FREE, BUT IS LESS LOADED THAN THIS ELEMENT
- TO GET THE OTHER ELEMENT, CAN NOTIFY THE SERVER TO ALLOCATE ELEMENT TO THE OTHER RANGE

16408

ELEMENT ASSIGNS IP(1)...IP(MIDDLE) TO ITSELF AND ASSIGNS IP(MIDDLE+1)...IP(N) TO FREE\_ELEMENT

<u>16410</u>

FUTURE REQUESTS FROM IP(MIDDLE+1)...IP(N) WILL BE REDIRECTED TO FREE\_ELEMENT AND THEIR REFERRER WILL BE TOLD TO REFER TO OTHER\_AGENT NEXT TIME

#### FIG. 166 – AGENT LOAD REDUCTION: REDUCING NUMBER OF IPS PER AGENT -IMPROVEMENTS

IMPROVEMENT 1: FINDING FREE ELEMENTS — ALSO POSSIBLE FOR THE SERVER TO PERIODICALLY LOOK FOR "UNDERLOADED" ELEMENTS (FOR EXAMPLE BY LOOKING FOR ELEMENTS THAT HAVE NOT BEEN OVERLOADED IN A WHILE), AND TO MERGE THEIR RANGES TO FREE SOME OF THEM UP FOR ASSIGNING TO OVERLOADED ELEMENTS

16602

IMPROVEMENT 2: INSTEAD OF LOGGING ALL THE IPS THAT ARE ACCESSING AN ELEMENT, THE ELEMENT CAN LOG ONLY THE X RECENT REQUESTS (X COULD BE 1,000 FOR EXAMPLE), OR KEEP A LOGARITHMIC RANDOM LIST OF X FROM THE PAST 24 HOURS, X/2 FROM PAST WEEK, X/4 FROM PAST MONTH, ETC.)

<u>16604</u>

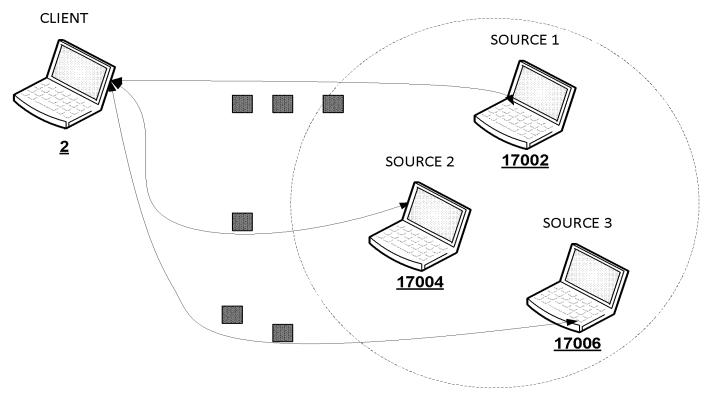
### FIG. 170 – MICRO CHUNKS – PRIOR ART – DIAGRAM

<u>17001</u>

FILE\_TO\_LOAD: THE FILE TO RETREIVE, COMPRISED OF 6 EQUALLY SIZED CHUNKS:



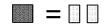
THE CLIENT LOADS THE FILE'S 6 CHUNKS FROM 3 SOURCES, TAKING MORE CHUNKS FROM THE FASTER SOURCES (3 FROM SOURCE\_1, 1 FROM SOURCE\_2, AND 2 FROM SOURCE\_3):

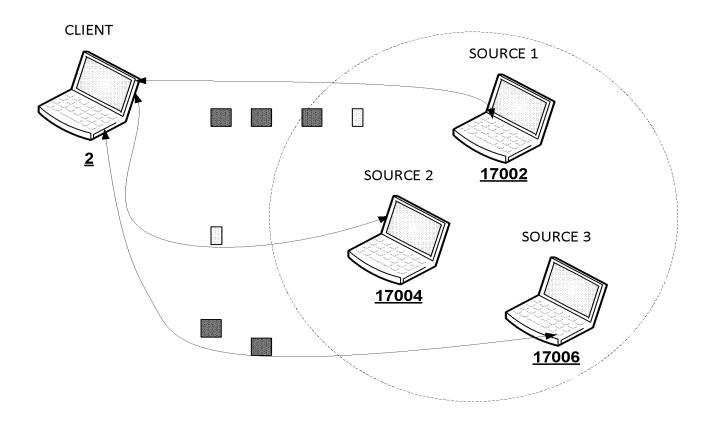


PREFERRED AGENTS (ONLINE & FASTER THAN THE OTHERS)

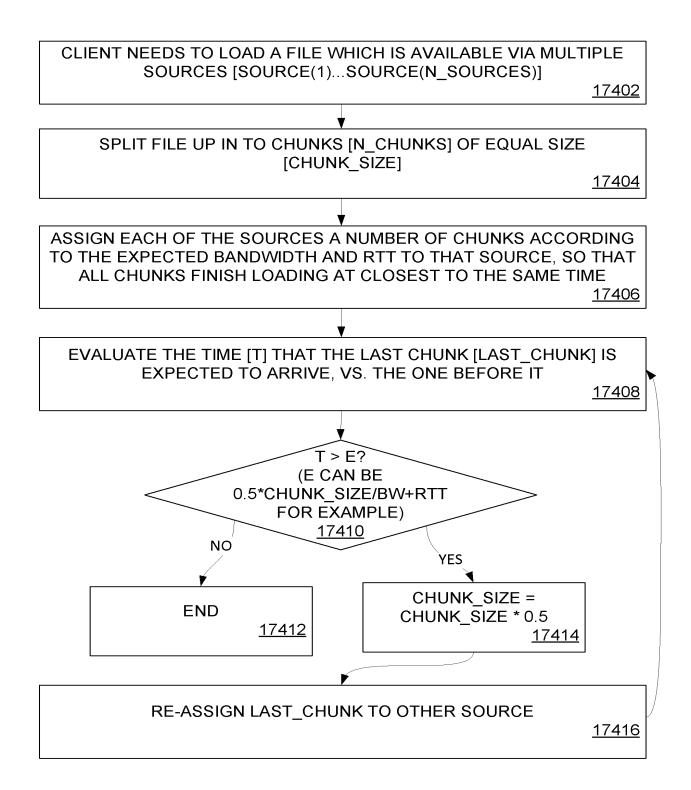
#### FIG. 172 - MICRO CHUNKS - DIAGRAM

THE ONE CHUNK FROM SOURCE\_2 IS SPLIT IN TO TWO SMALLER CHUNKS, IN ORDER NOT TO DELAY THE FULL LOAD:

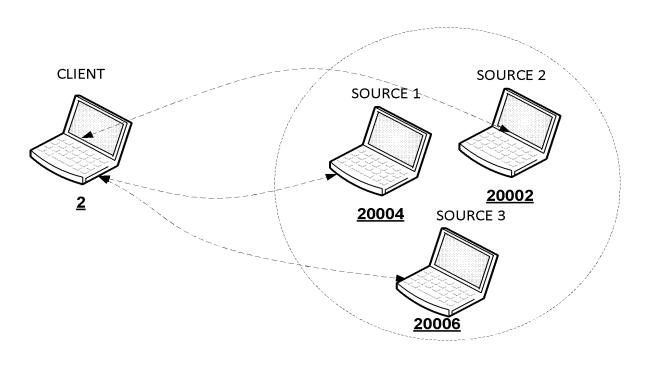




#### FIG. 174 – MICRO CHUNKS – FLOWCHART



#### FIG. 200 – LOWERING AVERAGE CONNECTION TIME IN MULTI-SOURCE SCENARIO - DIAGRAM





#### FIG. 202 – LOWERING AVERAGE CONNECTION TIME IN MULTI-SOURCE SCENARIO - FLOWCHART

RECEIVE THE LIST OF SOURCES THAT THE CLIENT CAN CONNECT TO (S1 ... SN)

20202

SOURCE = S1

20204

LOOK UP THE HISTORY OF CONNECTIONS TO SOURCE IN DATABASE (CAN BE DATABASE ON CLIENT, OR ON NETWORK, OR DISTRIBUTED). FOR EACH STAGE IN THE CONNECTION PROCESS (STAGE 1....STAGE N)

20206

FOR EACH STAGE IN CONNECTION (STAGE\_1...STAGE\_N), THE DATABASE WILL INCLUDE THE CONNECTION TIME FOR SOURCE FOR THAT STAGE (SOURCE\_T\_STAGE\_1 ... SOURCE\_T\_STAGE\_N), SIGNIFYING THE 90% OF THE CASES OF CONNECTION TO SOURCE 20208

IF SOURCE DOES NOT APPEAR IN THE DATABASE, ASSUME AVERAGE (SOURCE\_T\_STAGE\_1 ... SOURCE\_T\_STAGE\_N VALUES) FOR SOURCE. CAN BE BASED ON AVERAGE VALUES IN THE DATABASE

20210

STAGENUM = 1

20212

SET TIMER = SOURCE\_T\_STAGE\_STAGENUM

20214

DO "STAGE\_STAGENUM" CONNECTION TO SOURCE (E.G. AUTHENTICATION)

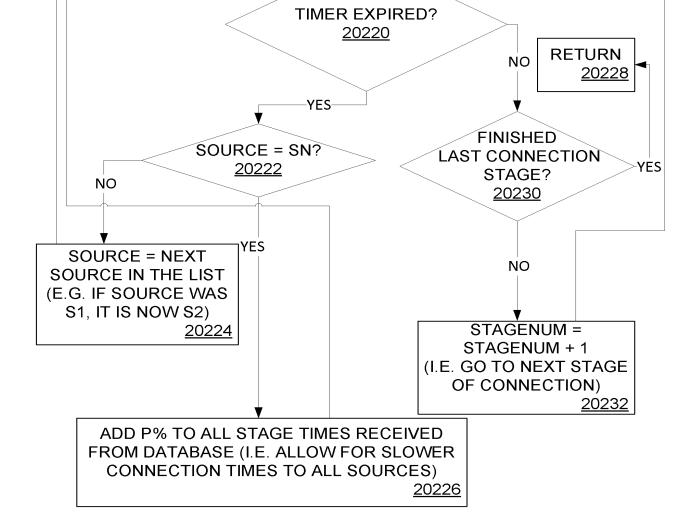
20216

WAIT UNTIL SUCCESSFUL CONNECTION OF STAGE, OR UNTIL TIMER EXPIRES

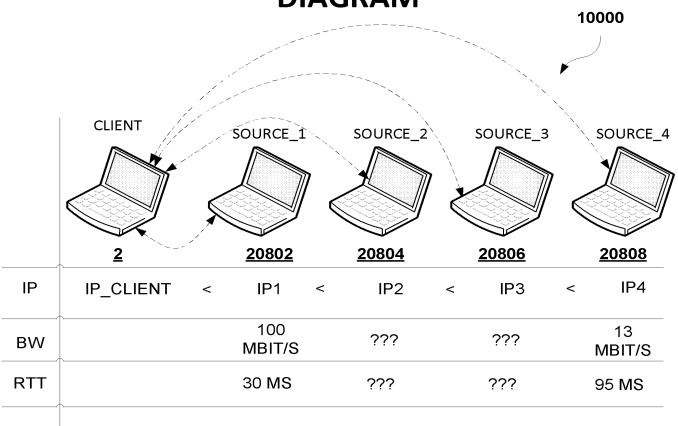
<u>20218</u>

NetNut Ltd. v. Bright Data Ltd. IPR2021-00465, EX. 2023 Page 46 of 121

TIMER EXPIRED?



## FIG. 208 – HEURISTICS FOR FINDING BW AND RTT TO NETWORK ELEMENTS – EXAMPLE - DIAGRAM



**CLIENT DATABASE** 

<u>20810</u>

CLIENT WANTS TO ASSES THE BW AND RTT TO SOURCE 2

20820

INFORMATION IN CLIENT'S DATABASE EXISTS ONLY TO SOURCE\_1 AND SOURCE\_4

20822

CLIENT USES BW AND RTT OF SOURCE\_1 AS BEST CASE, AND BW AND RTT OF SOURCE\_4 AS WORST CASE

20824

AFTER TRANSACTION, CLIENT UPDATES DATABASE WITH BW AND RTT OF SOURCE\_2 (IF DATA ALREADY EXISTS, CAN STORE MULTIPLE ENTRIES, USING LAST FEW ENTRIES' AVERAGE AS THE DATA)

#### FIG. 210 – HEURISTICS FOR FINDING BW AND RTT TO NETWORK ELEMENTS - FLOWCHART

CLIENT HAS DATABASE OF ALL NETWORK ELEMENTS THAT IT COMMUNICATED WITH, WITH THE BW AND RTT IN EACH OF THE PAST SEVERAL CONNECTIONS. NETWORK ELEMENTS ARE SORTED FROM ELEMENT\_1 TO ELEMENT\_N, BY THE IP\_DISTANCE (\*\*) BETWEEN THEM AND THE CLIENT (WHERE ELEMENT\_X IS CLOSER TO CLIENT THAN ELEMENT\_(X+1)

21002

CLIENT WANTS TO ASSES THE BW AND RTT TO SOURCE\_X

21004

INFORMATION IN CLIENT'S DATABASE EXISTS ONLY TO SOURCE\_L AND SOURCE\_M, WHERE L<X<M

21006

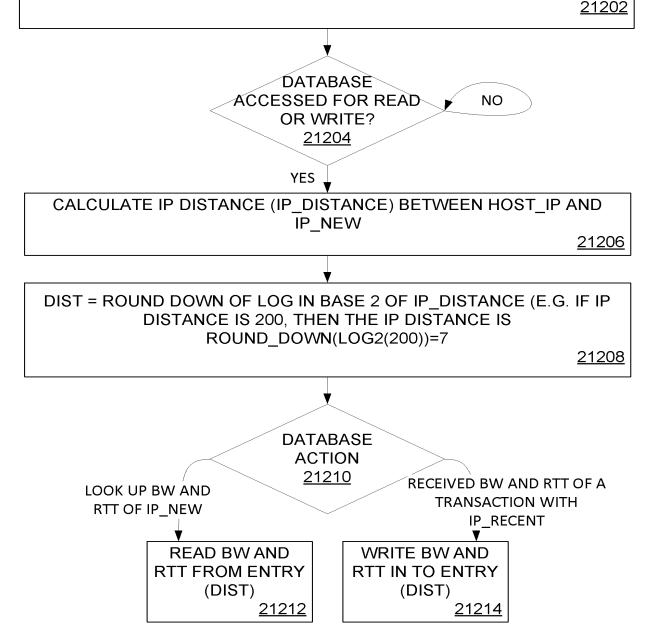
BEST CASE IS THE AVERAGE OF INFORMATION IN DATABASE (FOR BW AND RTT) FOR SOURCE\_L AND WORST CASE IS THE AVERAGE OF INFORMATION IN DATABASE (FOR BW AND RTT) FOR SOURCE\_M

21008

AFTER TRANSACTION, CLIENT UPDATES DATABASE WITH THE BW AND RTT OF SOURCE\_X FOR THIS CURRENT TRANSACTION (IF DATA ALREADY EXISTS, CAN STORE MULTIPLE ENTRIES, USING LAST FEW ENTRIES' AVERAGE AS THE DATA)

## FIG. 212 – HEURISTICS FOR FINDING BW AND RTT TO NETWORK ELEMENTS – MEMORY SAVING NOTATION

INITIALIZE – ON SYSTEM START: OPEN DATABASE THAT INCLUDES ENTRIES ONLY OF LOGARITHMIC IP DISTANCES FROM THE HOST MACHINE (I.E. FIRST ENTRY IS OF THE BW AND RTT OF THE HOSTS THAT ARE ONE IP DISTANCE AWAY (122.104.11.99 IS ONE IP DISTANCE FROM 122.104.11.100 AND FROM 122.104.11.98), THE SECOND ENTRY IS OF THE BW AND RTT OF THE HOSTS THAT ARE 2 IP DISTANCES AWAY, THEN 4 DISTANCES, THEN 8 DISTANCES, THEN 16 DISTANCES, ETC.)



## FIG. 214 – HEURISTICS FOR FINDING BW AND RTT TO NETWORK ELEMENTS – FASTER LEARNING NETWORK

SCHEDULE UPDATE OTHER NETWORK ELEMENTS WITH THIS NETWORK ELEMENT'S BW\_RTT TABLE. CAN BE DONE FOR EXAMPLE WHEN IDLE, OR WHEN FINISHED COMMUNICATING WITH ANOTHER ELEMENT, ETC

21404

UPDATE OTHER
NETWORK ELEMENTS

ACTION 21406 RECEIVE UPDATE LIST FROM OTHER NETWORK ELEMENT

CREATE A LIST OF WHICH NETWORK ELEMENTS TO UPDATE. CAN BE A LIST OF ALL NETWORK ELEMENTS COMMUNICATED WITH BY THIS NETWORK ELEMENT, OR BY GETTING A LIST FROM A CENTRAL SERVER, OR BY ONLY UPDATING THE ELEMENT THAT HAS JUST BEEN COMMUNICATED WITH, ETC, AND TRANSFER THE BW & RTT TABLE TO THEM

UPDATE\_DB IS
RECEIVED FROM
NETWORK ELEMENT,
WHO'S IP IS
IP\_ELEMENT AND
WHO'S IP DISTANCE
FROM LOCAL HOST IS
IP\_ELEMENT\_DIST

21408

<u>21410</u>

FOR EACH NON-EMPTY ENTRY (E) IN THE UPDATE\_DB, DO THE FOLLOWING: 21412

LOOKUP THE NETWORK ELEMENT THAT PROVIDED THE DATABASE IN THE LOCAL HOST'S DATABASE, AND STORE ITS BW AND RTT (AS BW HOST AND RTT HOST)

21414

LOOKUP BW AND RTT IN E, AND STORE AS BW\_E AND RTT\_E 21416

CALCULATE IP DISTANCE FROM IP\_ELEMENT TO IP IN E (IP\_DIST\_E) 21418

UPATE LOCAL HOST'S DATABASE ENTRY FOR ENTRY WITH DISTANCE (IP\_ELEMENT\_DIST + IP\_DIST\_E) WITH THE (BW\_HOST+BW\_E) AND (RTT\_HOST+RTT\_E) (OR OTHER METHOD)

<u>21420</u>

NetNut Ltd. v. Bright Data Ltd.

IPR2021-00465, EX. 2023 Page 51 of 121

## FIG. 216 – GEO IP. CREATING CONTIGUOUS RANGE, OPTIMAL FOR SPEED

BASED ON THE APPLICATION, DECIDE ON WHICH COST FUNCTION (COST) THE CONTIGUOUS RANGE IS TO BE BUILT BY (FOR EXAMPLE, WHERE COMMUNICATION SPEED IS REQUIRED, POSSIBLY USE PHYSICAL PROXIMITY, ETC)

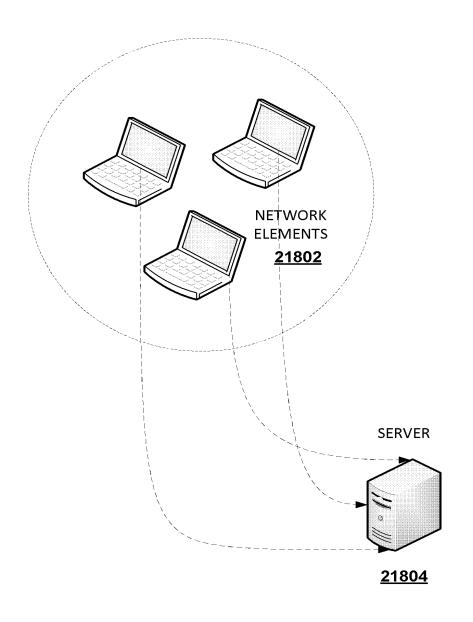
21602

V

FOR ALL IPS THAT EXIST ON THE PUBLIC NETWORK, FIND THE ORDER OF IPS FOR WHICH THE ORDER MAINTAINS THE FOLLOWING RELATIVE TO ANY OTHER ORDER:

 $(\sum_{i=1}^{n} O_i \cap COST(P(I), P(I+1)) == MIN)$ 

## FIG. 218 – FINDING OUT WHICH ELEMENTS IN A NETWORKED SYSTEM ARE ONLINE - DIAGRAM



# FIG. 220 – FINDING OUT WHICH ELEMENTS IN A NETWORKED SYSTEM ARE ONLINE – NETWORK ELEMENT FLOWCHART

(WITHIN NETWORK ELEMENT)

NETWORK ELEMENT PERIODICALLY PINGS THE SERVER (EVERY PING CYCLE TIME SECONDS. CAN BE 10 SECONDS FOR EXAMPLE)

22002

(WITHIN SERVER)

INITIALIZE DATABASE ON START: CREATE ENTRIES THAT CORRESPOND TO EACH OF THE NETWORK ELEMENTS IN THE NETWORK, CREATE AN 'ONLINE' BOOLEAN FIELD FOR EACH AND AN 'UPDATED' BOOLEAN FIELD

21404

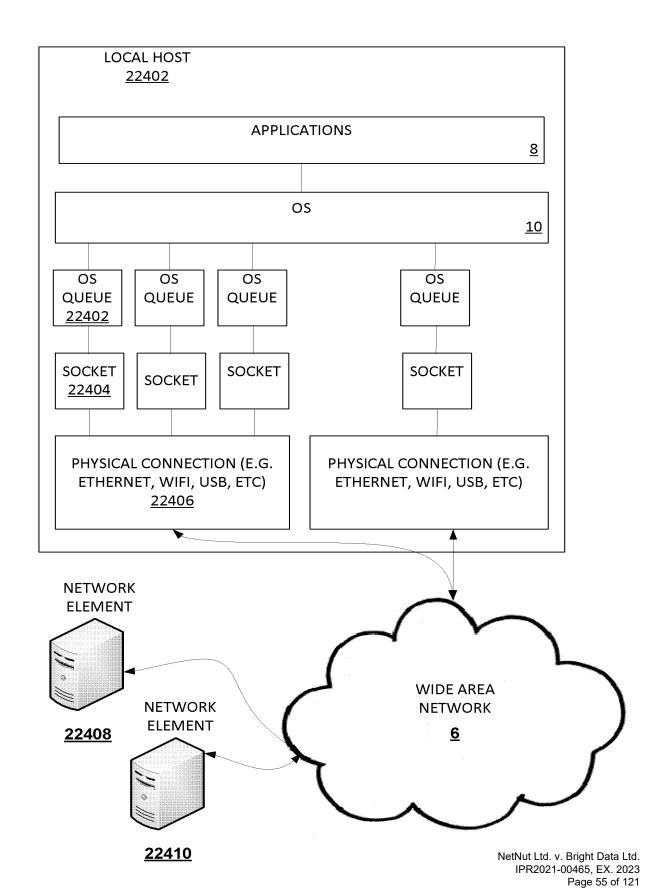
(WITHIN SERVER)

IF A PING IS RECEIVED FROM A NETWORK ELEMENT, UPDATE THE ENTRY IN THE SERVERS' DATABASE THAT CORRESPONDS TO THAT NETWORK ELEMENT TO 'ONLINE' (ONLINE=TRUE) AND MARK ITS 'UPDATED' FIELD AS 'T' 21406

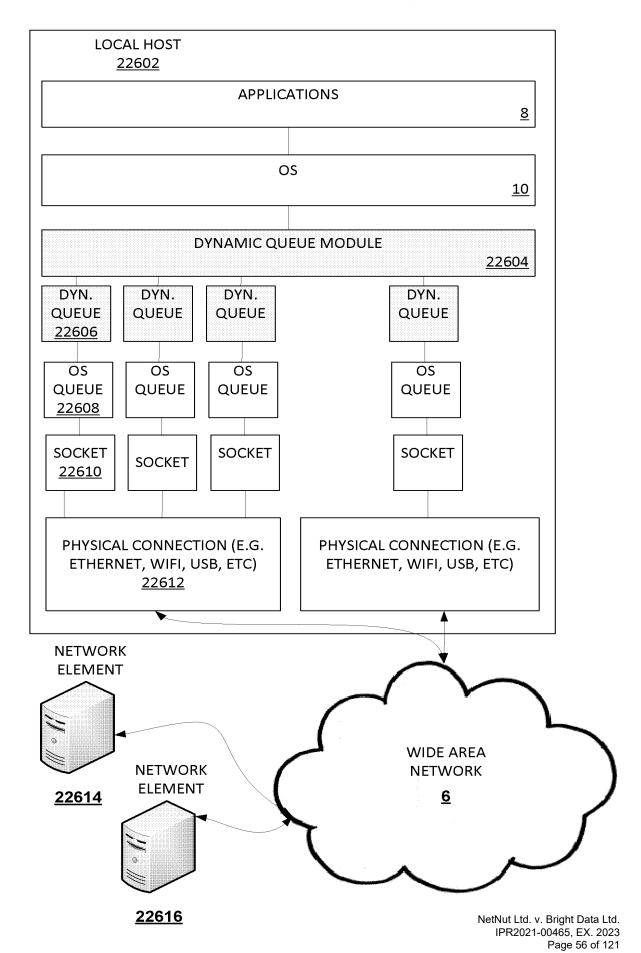
(WITHIN SERVER)

EVERY PING\_CYCLE\_TIME SECONDS, REVIEW THE DATABASE. FOR EACH ELEMENT IN WHICH ONLINE=TRUE, CHECK IF UPDATED=TRUE. IF IT IS — MARK ITS UPDATED=FALSE. IF IT ISN'T, MARK ITS ONLINE=FALSE

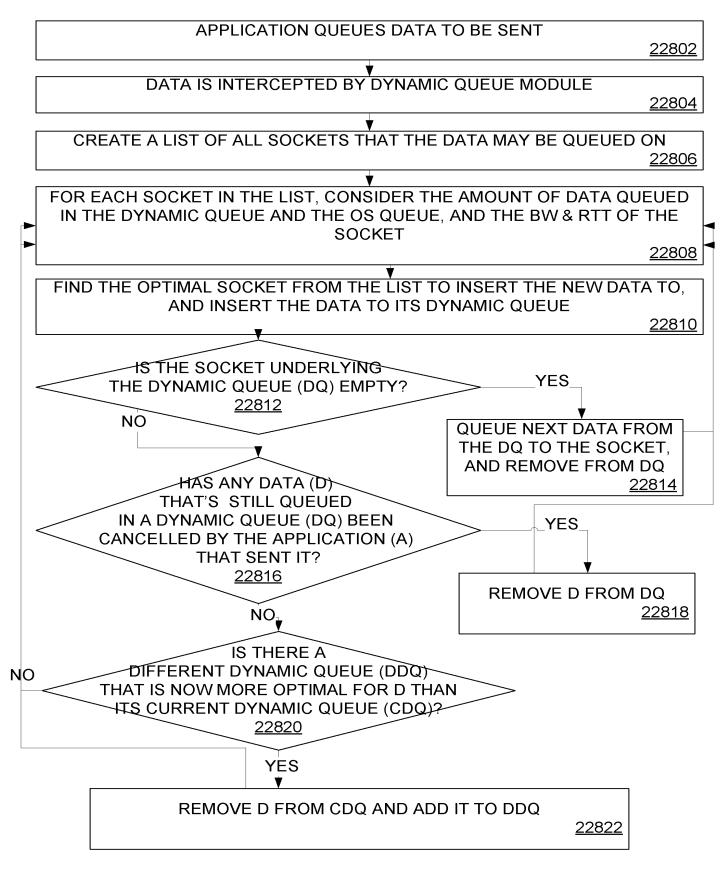
## FIG. 224 – DYNAMIC QUEUES FOR SENDING DATA – PRIOR ART



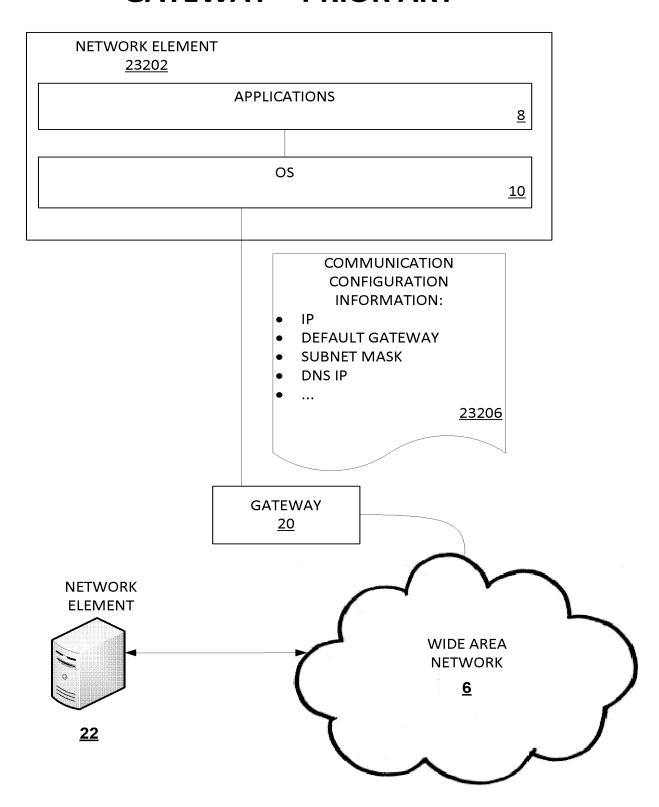
## FIG. 226 – DYNAMIC QUEUES FOR SENDING DATA – DIAGRAM



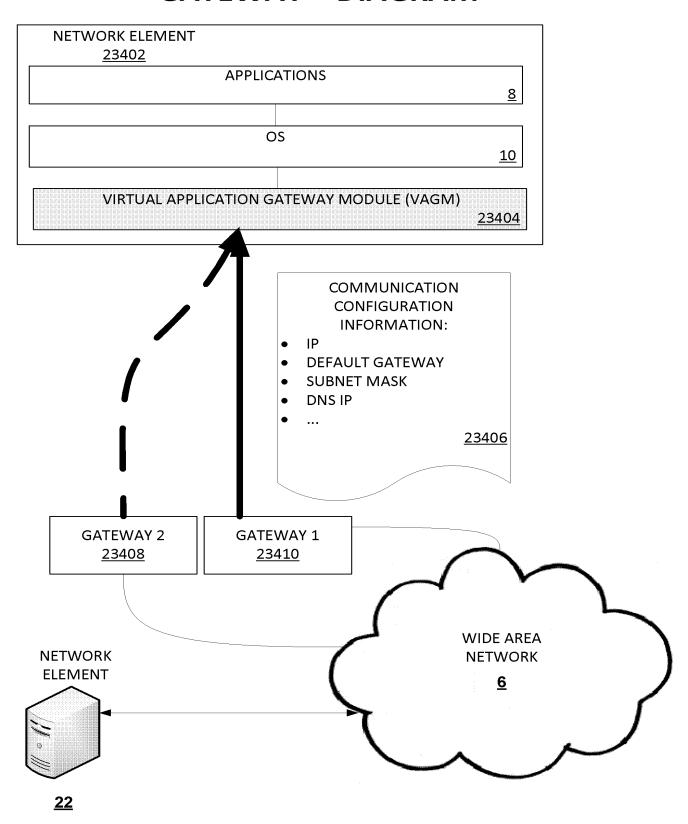
### FIG. 228 – DYNAMIC QUEUES FOR SENDING DATA – DIAGRAM



## FIG. 232 – VIRTUAL APPLICATION GATEWAY – PRIOR ART



## FIG. 234 – VIRTUAL APPLICATION GATEWAY – DIAGRAM

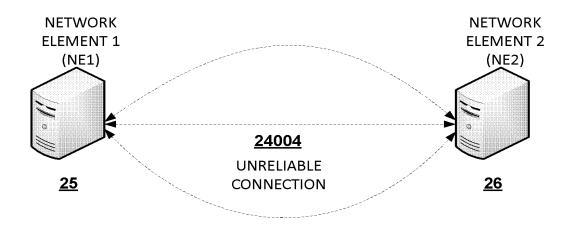


#### FIG. 236 – VIRTUAL APPLICATION GATEWAY – FLOWCHART

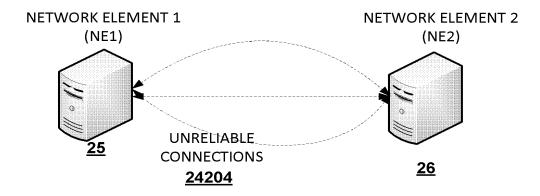
UPON CONNECTION TO THE NETWORK, REQUEST AND AN EXTERNAL IP FROM THE LOCAL GATEWAY AS THE OS WOULD DO NORMALLY 23602 SET UP TO TRAP REQUESTS FOR AQUIRING IP FOR THE NETWORK DEVICE FROM THE OS TOWARDS THE NETWORK 23604 HAS AN OS IP YES-REQUEST BEEN TRAPPED? 23608 PROVIDE A VALID LAN IP TO THE OS NO <u> 23610</u> VIRTUAL APPLICATION GATEWAY (VAG) ACTS AS A PROXY/NAT BETWEEN THE OS AND THE **NETWORK** <u>23606</u> DID THE VIRTUAL APPLICATION GATEWAY DISCONNECT FROM THE YES **NETWORK?** <u>23612</u> NO GET AN EXTERNAL IP FROM THE **LOCAL GATEWAY** 23614 YES RECEIVED AN IP? 23616 NO NO-**EXCEEDED** X MILLI SECONDS? ALLOW UP TO X MILLI SECONDS WITHOUT 23620 CONNECTIVITY 23618 SEND MSG TO OPERATING SYSTEM THAT YES-**NETWORK IS DISCONNECTED** 

> > Page 60 of 121

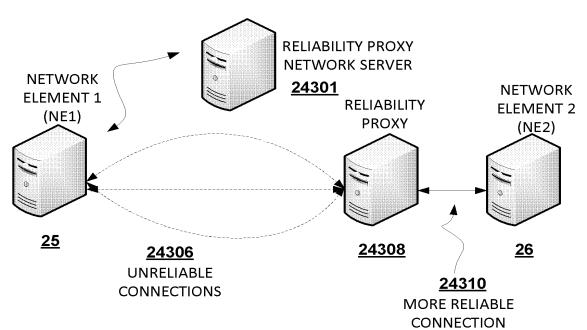
## FIG. 240 – RELIABLE CONNECTION PROXY – PRIOR ART



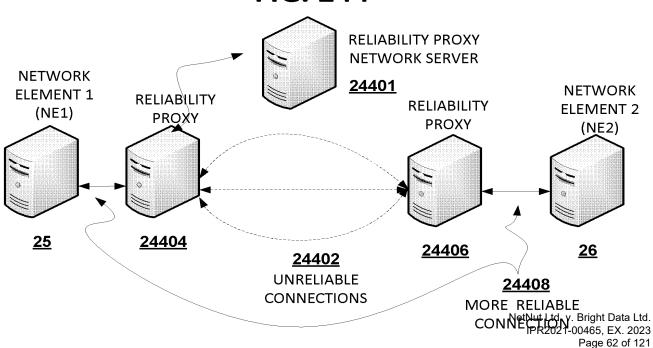
#### FIG. 242 - DIAGRAM



#### FIG. 243 - DIAGRAM



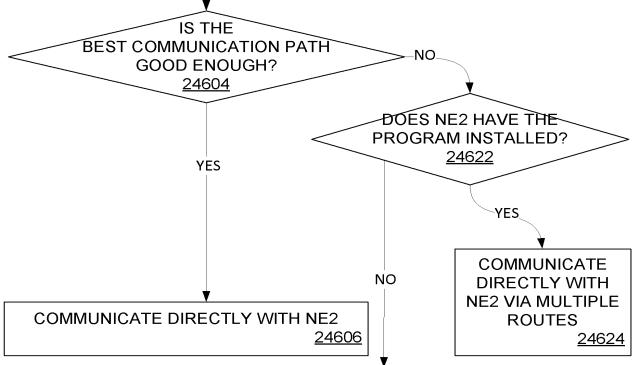




### FIG. 246 – RELIABLE CONNECTION PROXY – FLOWCHART

NETWORK ELEMENT 1 (NE1) NEEDS TO COMMUNICATE WITH NETWORK ELEMENT 2 (NE2)

24602



REQUEST RELIABILITY PROXY FROM RELIABILITY PROXY NETWORK SERVER (RPNS) 24608

RELIABILITY SERVER PROVIDES BACKA RELIABILITY PROXY DEVICE (RPD), WHO'S PATH TO NE2 IS MORE RELIABLE THAN A DIRECT CONNECTION FROM NE1 TO NE2 (IF EXISTS SUCH A PATH)

24610

NE1 SENDS INFORMATION TO RPD (THROUGH THE MULTIPLE ROUTES) INSTEAD OF TO NE2, AND RPD PROXIES THE INFORMATION TO NE2

### FIG. 247 -TRANSMITTING PACKETS OVER MULTIPLE ROUTES

GET FROM ROUTING UNIT THE NUMBER OF ROUTES AVAILABLE, AND THE RELIABILITY OF EACH ONE

♥

24702

BASED ON THE DESIRED RELIABILITY (DR), DETERMINE WHICH ROUTES TO USE FOR THIS COMMUNICATION. DO THIS BY ORDERING THE ROUTES FROM THE HIGHEST RELIABILITY (R1) TO THE LOWEST RELIABILITY (RN), AND MULTIPLY (1-R1)\*(1-R2)\*.... UNTIL THIS PRODUCT IS LOWER THAN (1-DR). THESE ARE THE ROUTES TO USE.

24704

FOR EACH PACKET TO BE TRANSMITTED, TRANSMIT IT OVER ALL CHOSEN ROUTES

24706

FOR EACH UNIQUE PACKET OF COMMUNICATION RECEIVED, SEND IT TO THE APPLICATION, AND DISCARD THE OTHER SIMILAR PACKETS RECEIVED FROM THE OTHER ROUTES

### FIG. 248 – RELIABLE CONNECTION PROXY – FLOWCHART

NETWORK ELEMENT 1 (NE1) NEEDS TO COMMUNICATE WITH NETWORK ELEMENT 2 (NE2) AND TO KEEP THE CONNECTION ALIVE

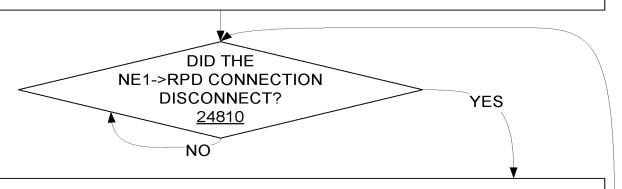
24802

CONNECTION FROM NE1 TO NE2 IS ESTABLISHED THROUGH THE VIRTUAL APPLICATION GATEWAY IN EACH OF THEM IF EXISTS

24804

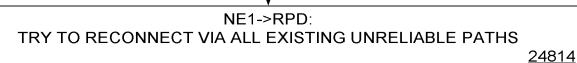
IF THE VAG DOES NOT EXIST IN EITHER OF THEM, THEN A RELIABILITY PROXY (RP) IS USED: IF THE VAG DOES NOT EXIST IN NE1, THEN IT IS CONFIGURED TO USE AN RP, AND IF IT DOES NOT EXIST IN NE2 THEN NE1 (OR ITS RP) REQUEST A RP FOR NE2 FROM THE SERVER

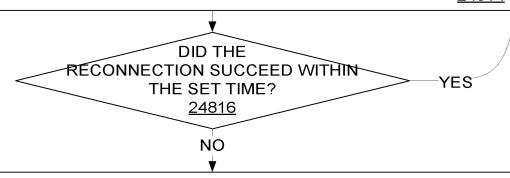
24808



THE RELIABILITY PROXIES (OR THE VAGS IN NE1 AND OR IN NE2) HOLD THE CONNECTION BETWEEN THEM FOR A SET SHORT AMOUNT OF TIME, TO ENABLE RECONNECTION

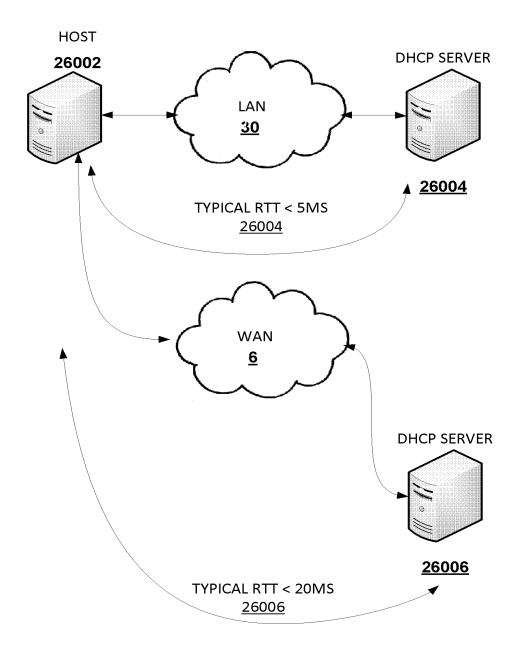
<u>24812</u>



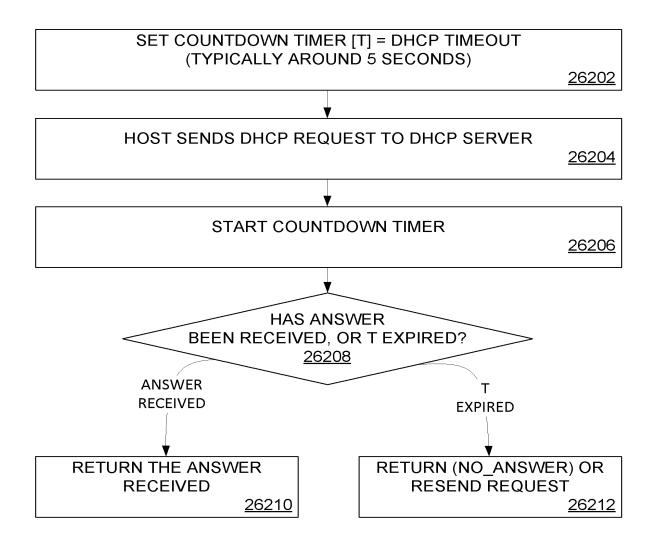


NOTIFY THE OS OF THE DISCONNECTION

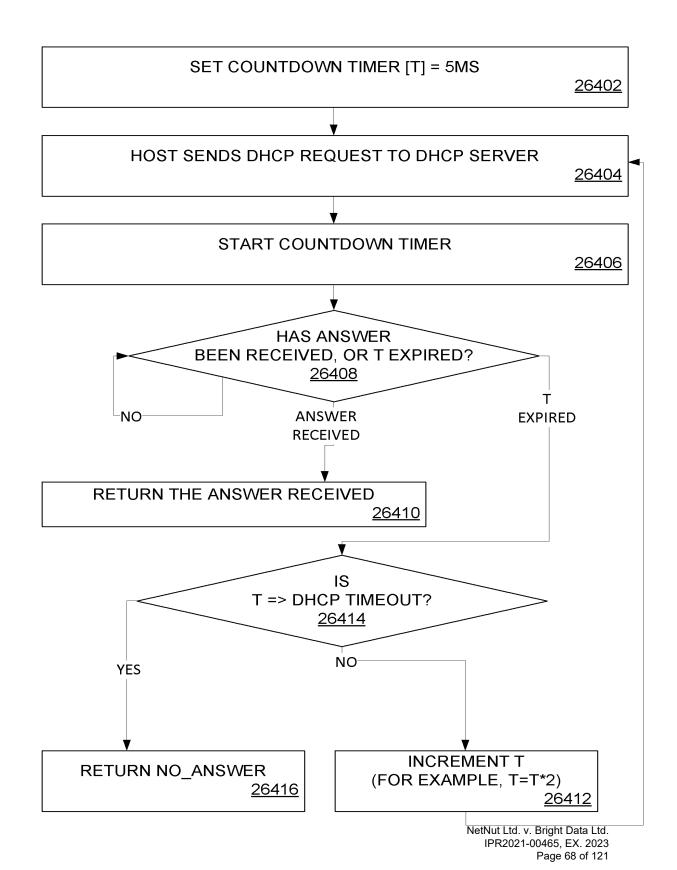
## FIG. 260 – AGGRESSIVE DHCP – PRIOR ART DIAGRAM



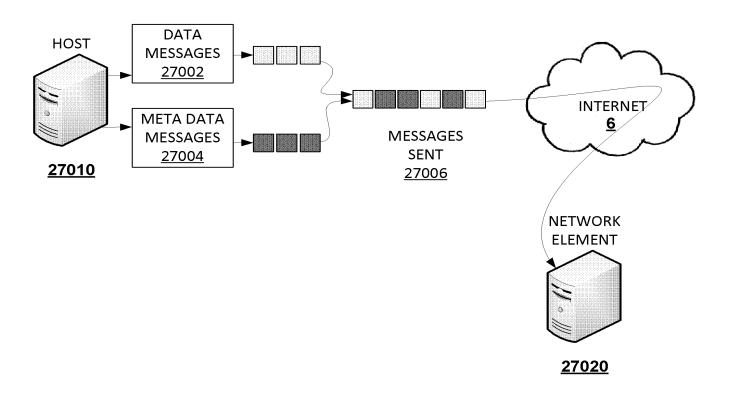
### FIG. 262 – AGGRESSIVE DHCP – PRIOR ART FLOWCHART



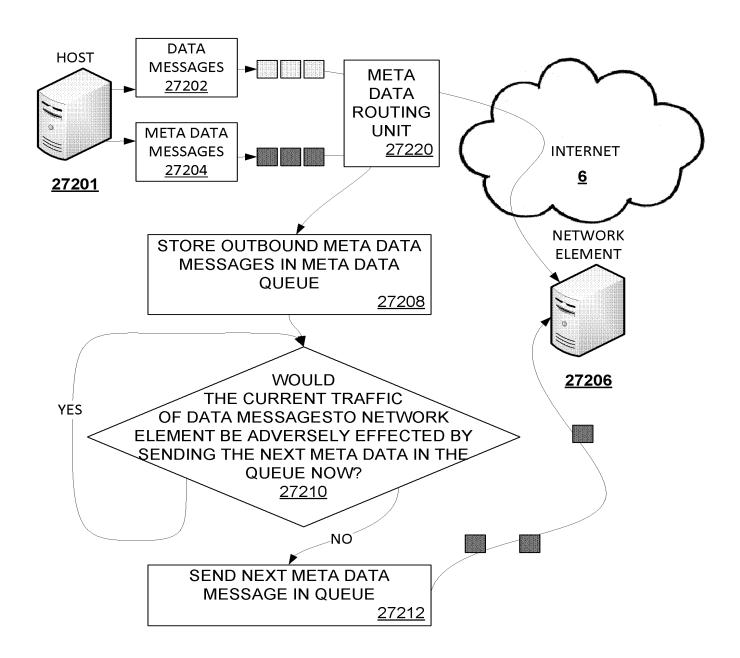
## FIG. 264 – AGGRESSIVE DHCP – PRIOR ART FLOWCHART



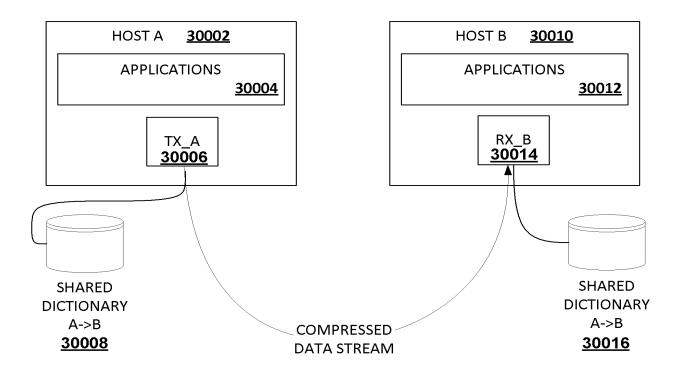
#### FIG. 270 – DATA / META DATA PRIORITIZATION – PRIOR ART – DIAGRAM



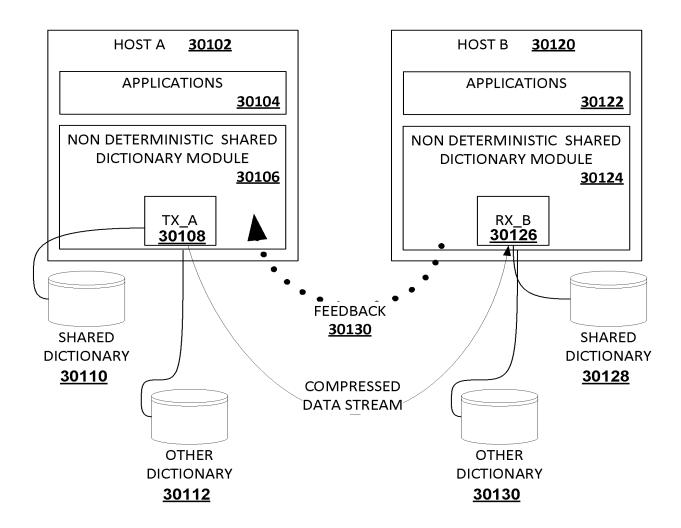
## FIG. 272 – DATA / META DATA PRIORITIZATION – DIAGRAM



## FIG. 300 – COMPRESSION WITH NON DETERMINISTIC SHARED DICTIONARY – PRIOR ART DIAGRAM

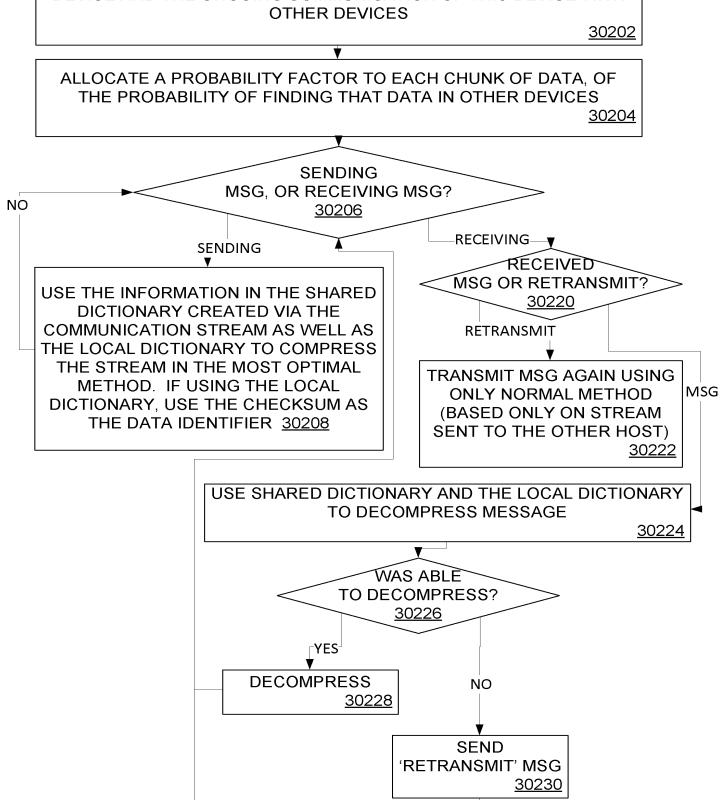


## FIG. 301 – COMPRESSION WITH NON DETERMINISTIC SHARED DICTIONARY – DIAGRAM



# FIG. 302 – COMPRESSION WITH NON DETERMINISTIC SHARED DICTIONARY – FLOWCHART

ALLOCATE A LOCAL BUFFER FOR THE DICTIONARY, AND BUILD LOCAL DICTIONARY FOR LOCAL DEVICE BASED ON INFORMATION IN THE DEVICE AND THE ONGOING COMMUNICATION OF THIS DEVICE WITH OTHER DEVICES

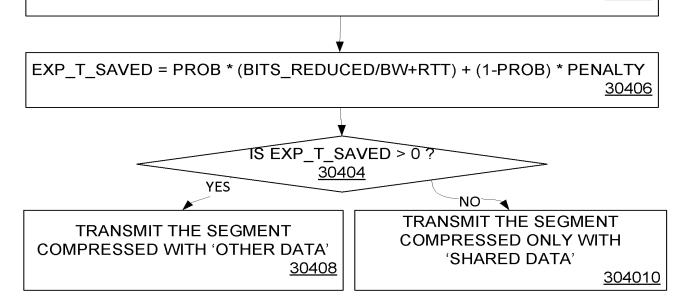


NetNut Ltd. v. Bright Data Ltd. IPR2021-00465, EX. 2023 Page 73 of 121

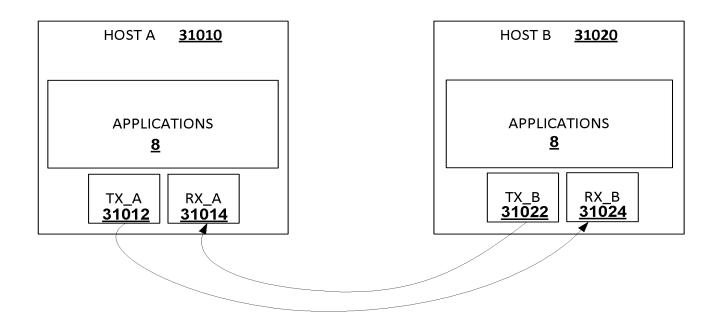
# FIG. 304 – COMPRESSION WITH NON DETERMINISTIC SHARED DICTIONARY – IMPROVEMENT

#### **SET PARAMETERS:**

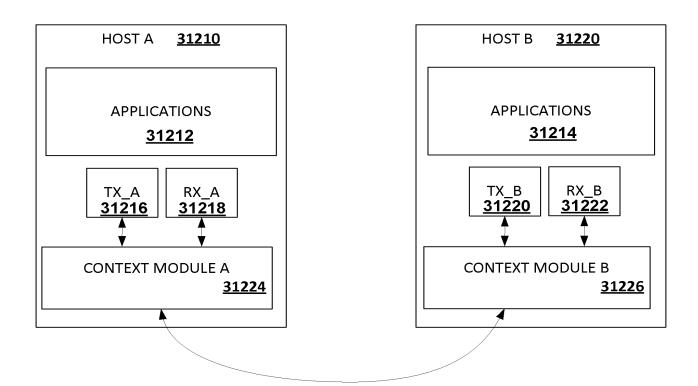
- BW=UPSTREAM BW TO DESTINATION
- SEGMENT\_COMPRESSED = SEGMENT SIZE WHEN COMPRESSED WITH SHARED DICTIONARY ONLY
- SEGMENT\_COMPRESSED\_WITH\_OTHER = SEGMENT SIZE WHEN COMPRESSED WITH THE 'OTHER DATA'
- BITS\_REDUCED = SEGMENT\_COMPRESSED SEGMENT\_COMPRESSED\_WITH\_OTHER
- PROB = ESTIMATED PROBABILITY THAT DESTINATION HAS THE RELEVANT 'OTHER DATA'
- PENALTY = (SEGMENT\_COMPRESSED\_WITH\_OTHER/BW + RTT) + RTT 30402



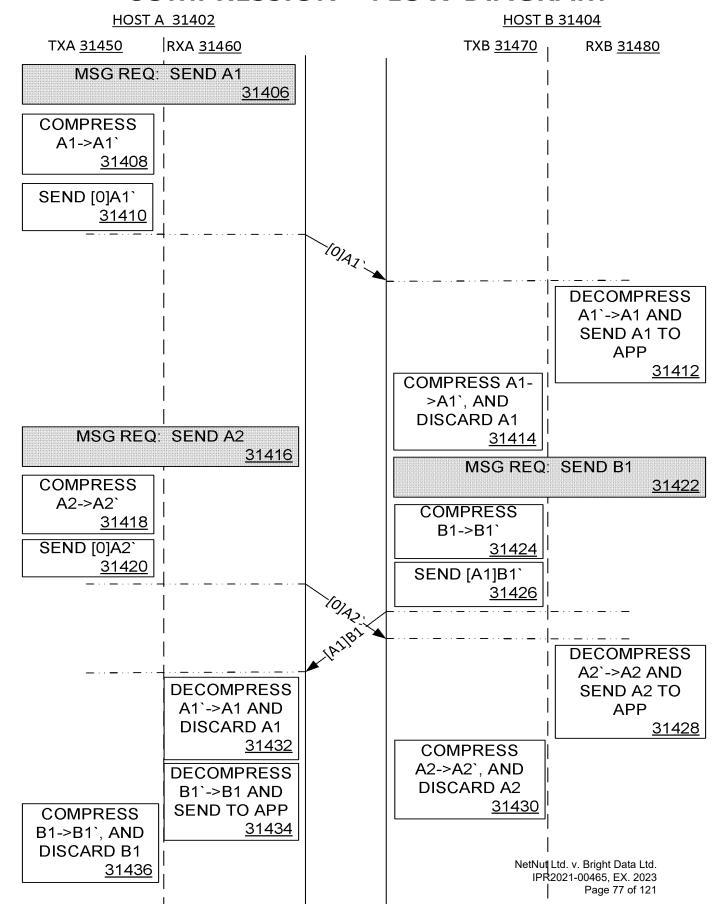
# FIG. 310 – TWO WAY STREAM COMPRESSION – PRIOR ART DIAGRAM

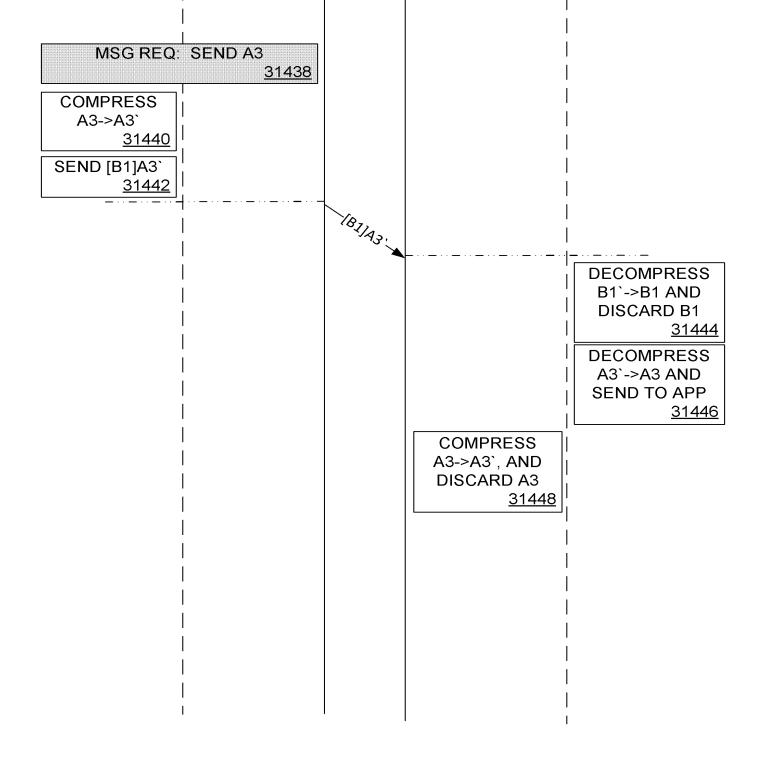


## FIG. 312 – TWO WAY STREAM COMPRESSION – DIAGRAM



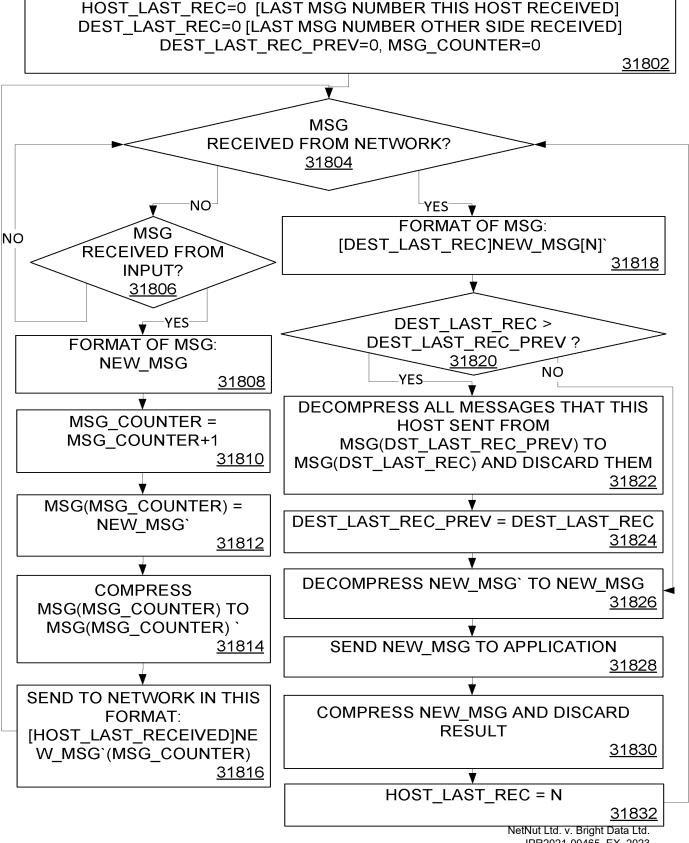
## FIG. 314 – TWO WAY STREAM COMPRESSION – FLOW DIAGRAM





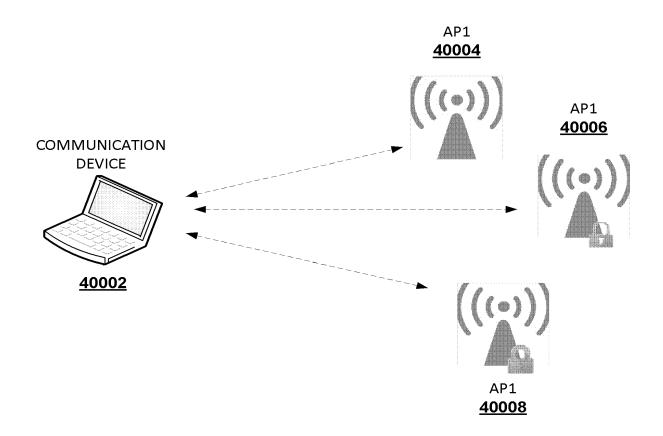
## FIG. 318 – TWO WAY STREAM COMPRESSION – FLOW CHART

INIT:



etNut Ltd. v. Bright Data Ltd. IPR2021-00465, EX. 2023 Page 79 of 121

#### FIG. 400 – ACCESS POINT SETUP



## FIG. 402 – GUESSING PASSWORDS FOR LOCKED ACCESS POINTS - FLOWCHART

IF TRYING TO CONNECT TO A LOCKED ACCESS POINT FOR WHICH HOST DOES NOT HAVE CORRECT PASSWORD, TRY THE FOLLOWING:

40202

TRY THE MOST COMMON PASSWORDS FOR THE LOCAL GEOGRAPHY

40204

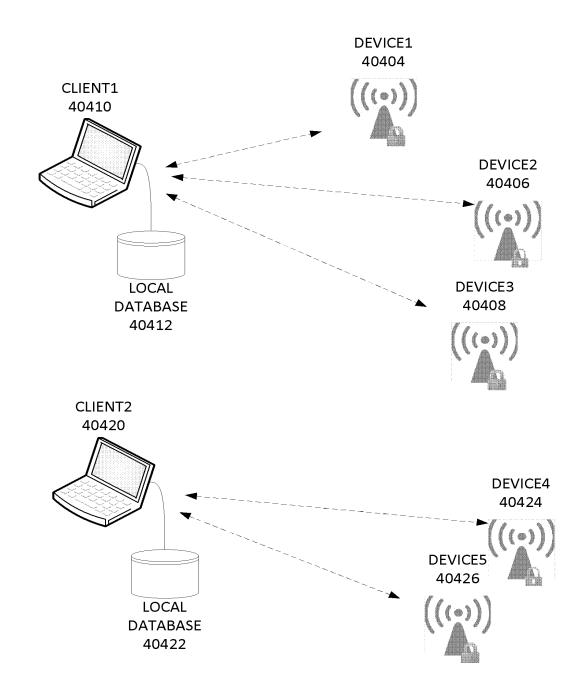
TRY PASSWORDS THAT THIS DEVICE USED TO CONNECT TO OTHER DEVICES IN THE PAST

<u>40206</u>

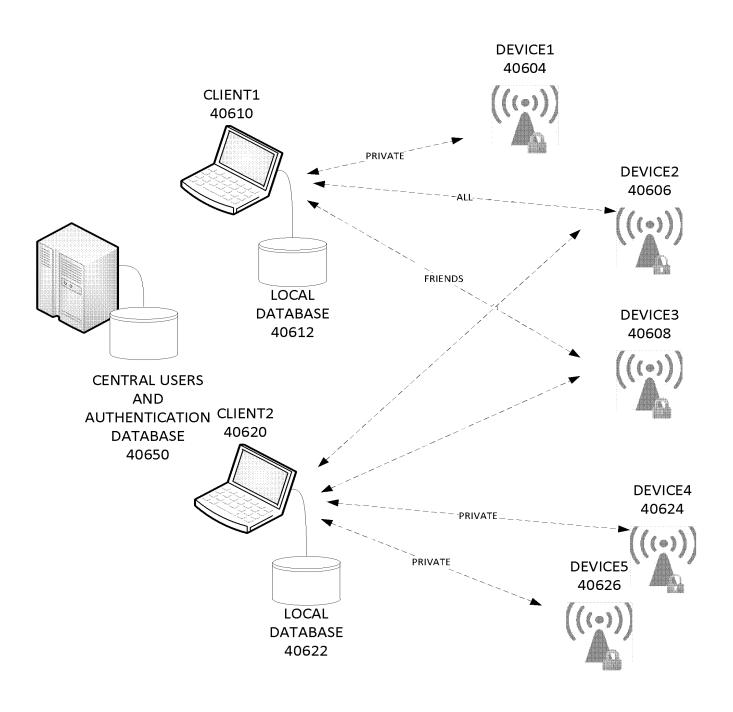
TRY THE DEFAULT PASSWORDS USED BY THAT DEVICE'S MANUFACTURER

40208

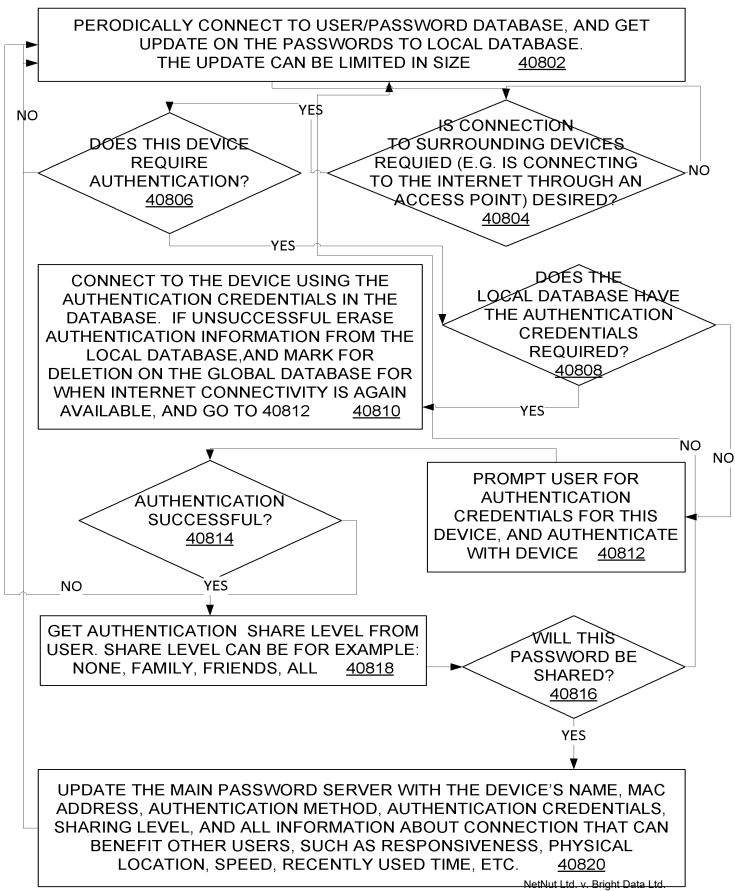
## FIG. 404 – PASSWORD SHARING – PRIOR ART - DIAGRAM



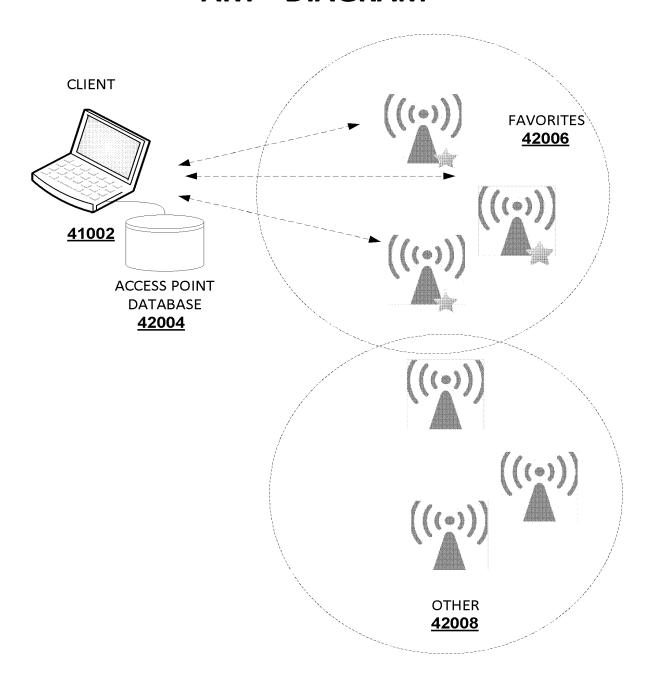
## FIG. 406 – PASSWORD SHARING – DIAGRAM



## FIG. 408 – PASSWORD SHARING - FLOWCHART



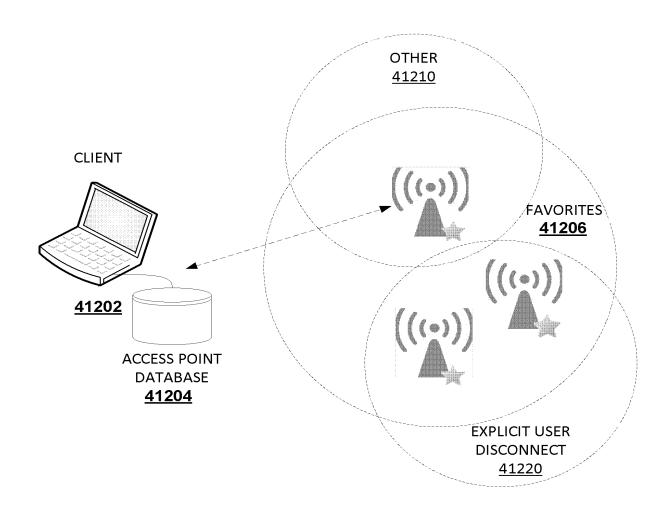
## FIG. 410 – SMART CUTOFF – PRIOR ART - DIAGRAM



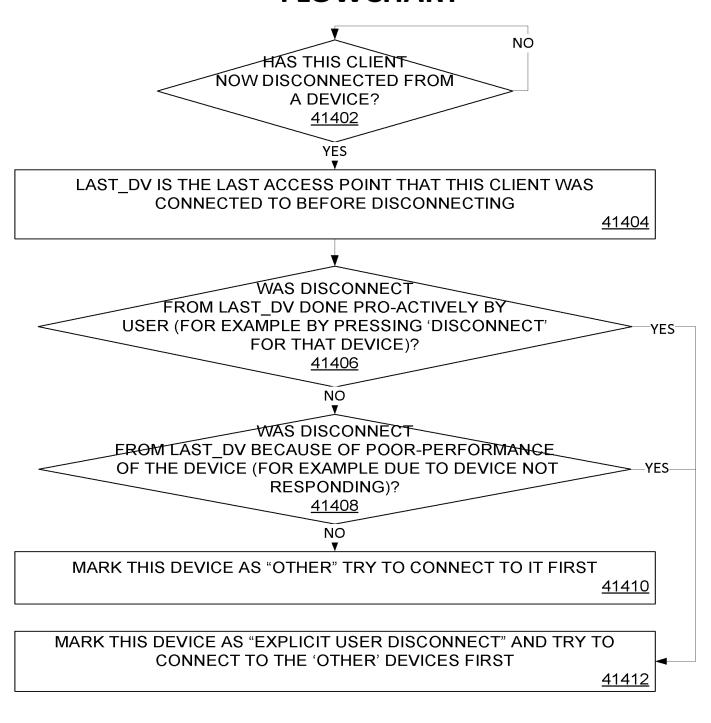
WHEN CLIENT IS STARTED UP, IT CONNECTS TO ONE OF THE 'FAVORITE' ACCESS POINTS, TYPICALLY SORTED BY SIGNAL STRENGTH

9202

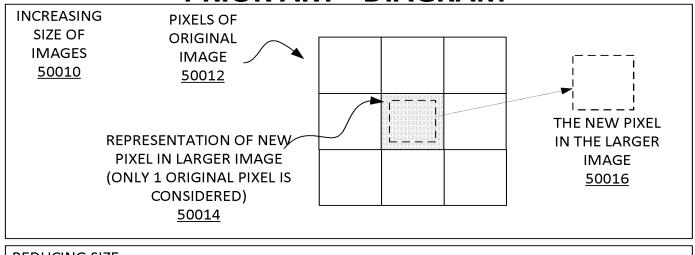
#### FIG. 412 - SMART CUTOFF - DIAGRAM

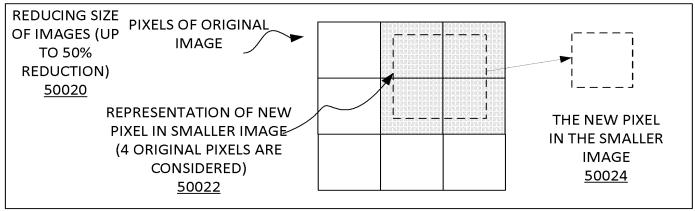


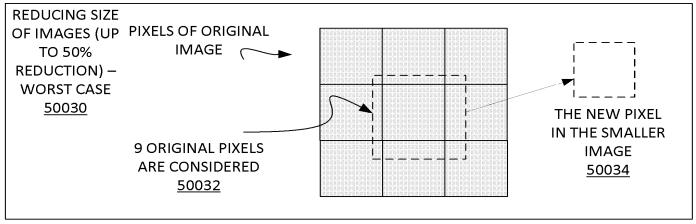
## FIG. 414 – SMART CUTOFF – FLOWCHART

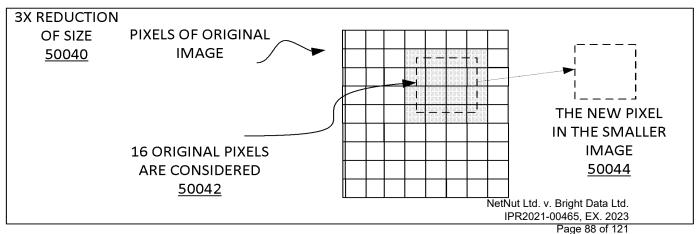


## FIG. 500 – QUICK IMAGE RESIZING – PRIOR ART - DIAGRAM







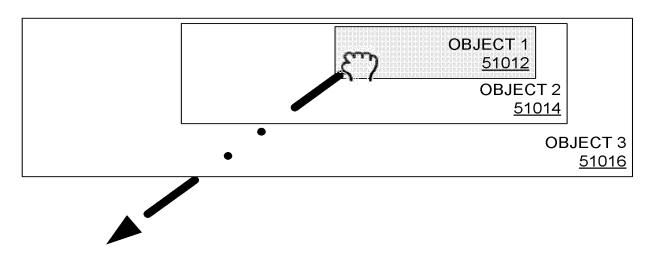


## FIG. 504 – QUICK IMAGE RESIZING – FLOWCHART

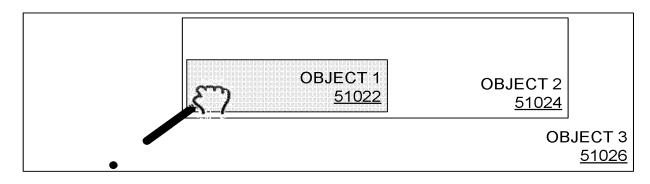
GET ORIGINAL IMAGE IN PIXEL FORM [IMG(0)], AND SCALE OF THE RESIZE [RESIZE] IN % (RESIZE IS THE SIZE OF THE TARGET IMAGE RELATIVE TO THE SOURCE IMAGE. FOR EXAMPLE, IF THE NEW IMAGE IS TO BE 4X SMALLER, THEN THE RESIZE IS 1/4X=25%) 50402 N=1<u>50404</u> RESIZE >50%? 50406 -YES RESIZE IMG[N-1] IN PRIOR **ART METHOD** NO 50408 CREATE NEW IMAGE [IMG(N)], BY AVERAGING EVERY 4 PIXELS IN IMG(N-1)TO ONE PIXEL IN IMG(N) 50410 RESIZE = RESIZE \* 2 (E.G. IF RESIZE WAS 20%, THEN NOW IT IS 40%), AND N=N+1 50412

### FIG. 510 – OBJECT MOVEMENT HEREDITY – PRIOR ART - DIAGRAM

START OF DRAG 51010



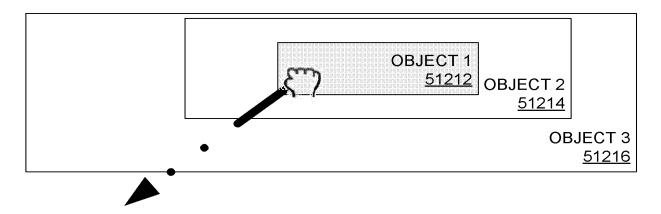
END OF DRAG 51020



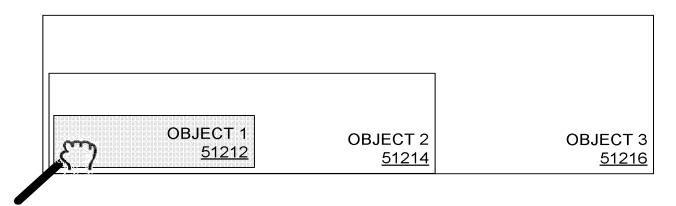


### FIG. 512 – OBJECT MOVEMENT HEREDITY – DIAGRAM

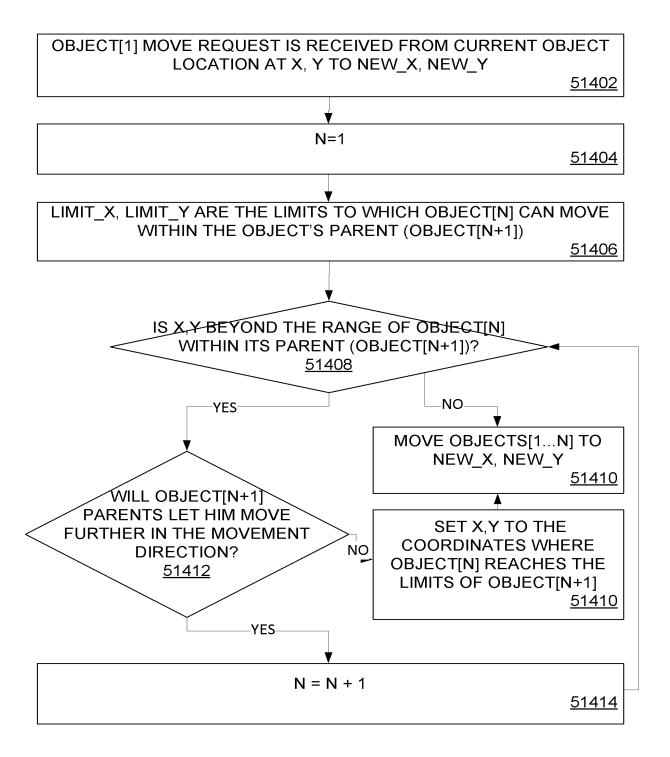
START OF DRAG 51210



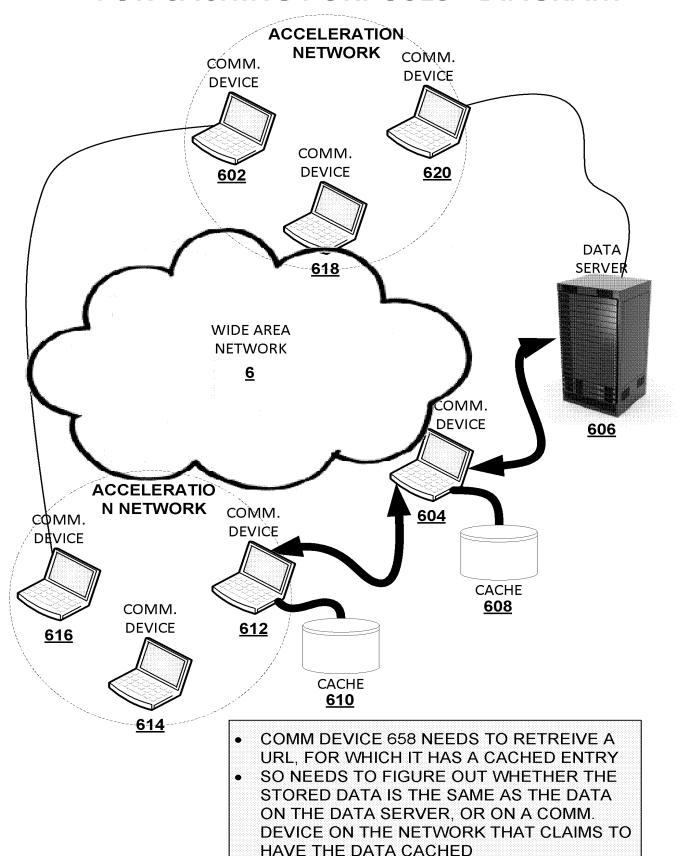
END OF DRAG 51220



## FIG. 514 – OBJECT MOVEMENT HEREDITY – FLOWCHART



## FIG. 600 – SAMENESS BETWEEN URLS FOR CACHING PURPOSES - DIAGRAM

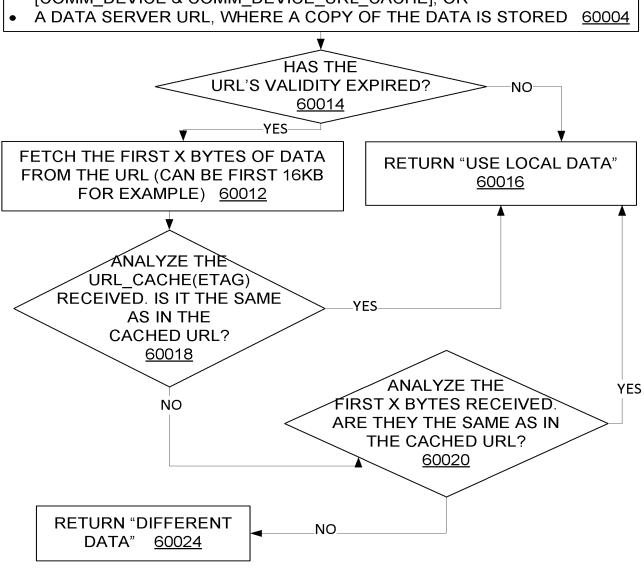


## FIG. 602 — SAMENESS BETWEEN URLS FOR CACHING PURPOSES - FLOWCHART

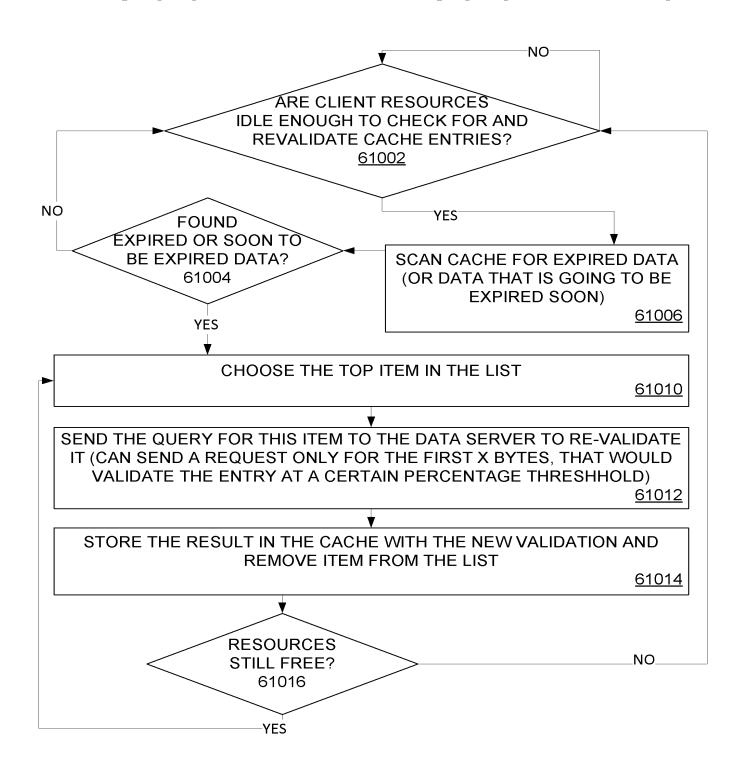
MODULE RECEIVES URL [URL] AND A CACHED ENTRY OF THAT URL WHICH IS EXPIRED 60002

MODULE RECEIVES A SOURCE OF THE URL THAT HAS A VALID COPY STORED – EITHER:

 A COMMUNICATION DEVICE THAT HAS THE URL'S CACHE DATA ENTRY [COMM\_DEVICE & COMM\_DEVICE\_URL\_CACHE], OR



#### FIG. 610 – REVALIDATING CACHE ENTRIES



### FIG. 620 – NDFS CACHE OVERWRITE REDUCTION AND NDFS CACHE CLEANUP – PRIOR ART

STORAGE DEVICE:

 OS DATA
 CACHE DATA
 FREE SPACE

 62002
 62004
 62006

----

FILE SYSTEM WRITE DIRECTION 62020

CACHE SYSTEM WRITE DIRECTION 62030

### FIG. 622 – NDFS CACHE OVERWRITE REDUCTION AND NDFS CACHE CLEANUP – DIAGRAM

STORAGE DEVICE:

OS DATA FREE SPACE CACHE DATA 62012 62014 62016

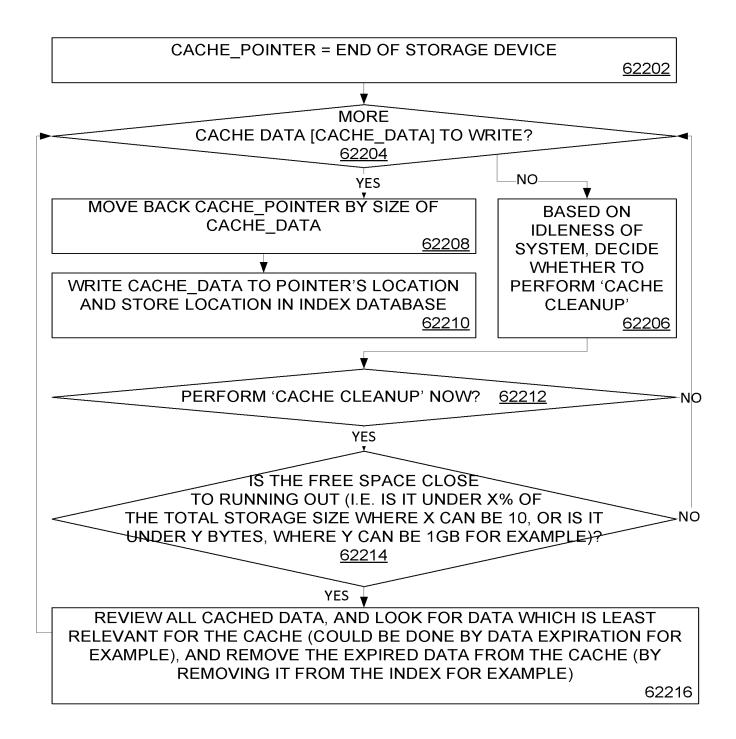
----

**—** 

FILE SYSTEM WRITE DIRECTION 62040

CACHE SYSTEM WRITE DIRECTION 62050

### FIG. 622 – NDFS CACHE OVERWRITE REDUCTION AND NDFS CACHE CLEANUP – FLOWCHART



## FIG. 630 – NDFS CACHE OVERWRITE REDUCTION – DIAGRAM

#### STORAGE DEVICE:

OS DATA 63002	FREE SPACE 63004	60:010:010:010:010:010:010:010:010:010:0	CACHE CHUNK 1 63008	FREE SPACE 63010	CACHE CHUNK 2 63012	CACHE CHUNK 1 63014
---------------------	------------------------	--	------------------------------	------------------------	------------------------------	------------------------------



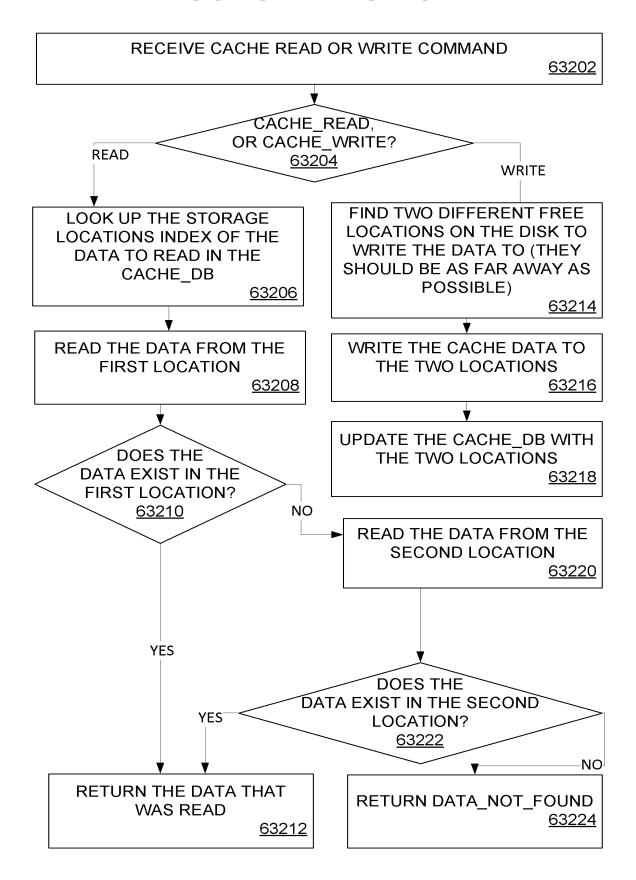
FILE SYSTEM WRITE DIRECTION 63020



CACHE SYSTEM SECONDARY WRITE DIRECTION 63030



## FIG. 632 – NDFS CACHE OVERWRITE REDUCTION – FLOWCHART



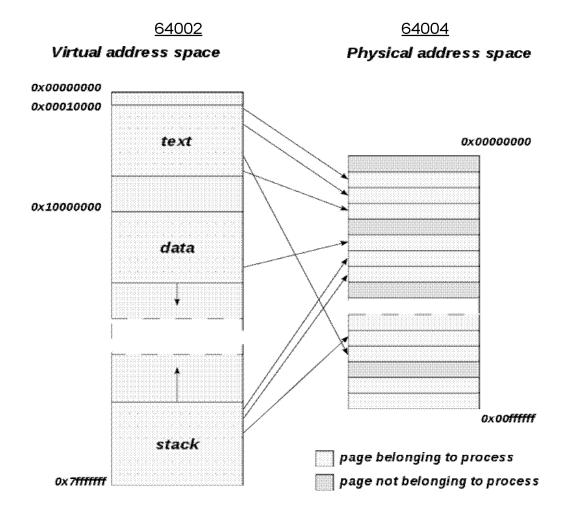
## FIG. 634 – NDFS CACHE OVERWRITE REDUCTION – NUMBER OF CHUNK COPIES TO STORE

#### STORAGE DEVICE:

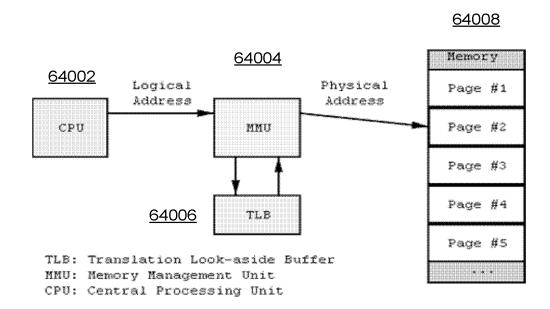
	CACHE CHUNK 2 63408	CACHE CHUNK 1 63410	FREE SPACE <u>63412</u>	CACHE CHUNK 2 <u>63414</u>	CACHE CHUNK 1 63416
--	------------------------------	------------------------------	-------------------------------	-------------------------------------	------------------------------

N<sup>TH</sup> COPY OF CHUNKS 63420 2<sup>ND</sup> COPY OF CHUNKS 63430 1<sup>ST</sup> COPY OF CHUNKS 63440

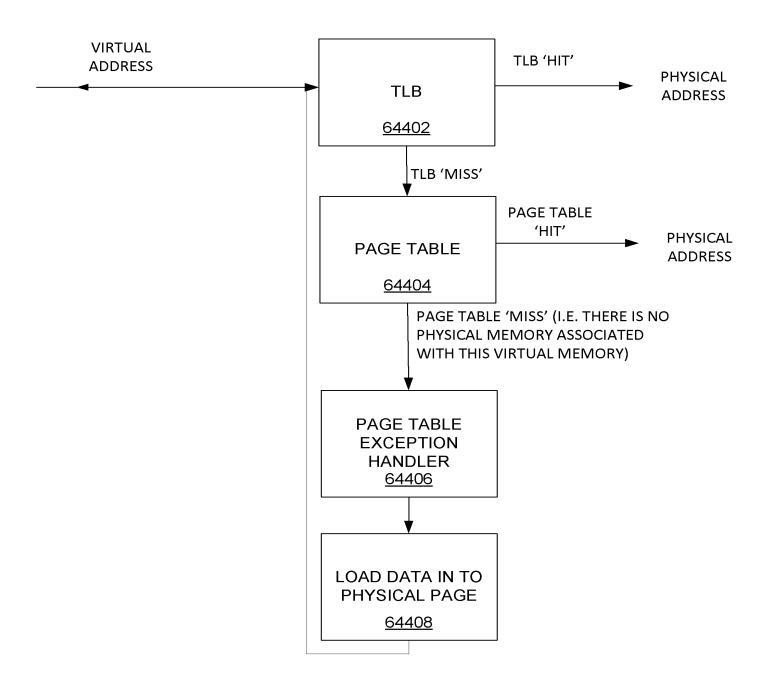
# FIG. 640 – NDCACHE – PRIOR ART PHYSICAL ADDRESS VS. VIRTUAL ADDRESS



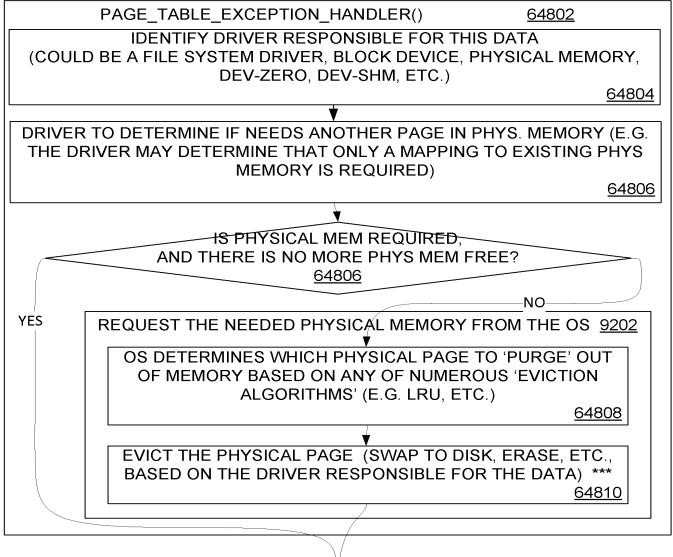
# FIG. 642 – NDCACHE – MMU – MEMORY MANAGEMENT UNIT – TRANSLATION BETWEEN VIRTUAL ADDRESS AND PHYSICAL ADDRESS



## FIG. 644 – NDCACHE – PRIOR ART – MMU OPERATION



## FIG. 648 – NDCACHE – PRIOR ART – EXCEPTION HANDLER



LOAD DATA INTO PHYSICAL PAGE 64812

DRIVER DETERMINES THE METHOD BY WHICH TO LOAD THE DATA IN TO THE PHYSICAL MEMORY OR TO OTHERWISE MAP THE VIRTUAL ADDRESS
64814

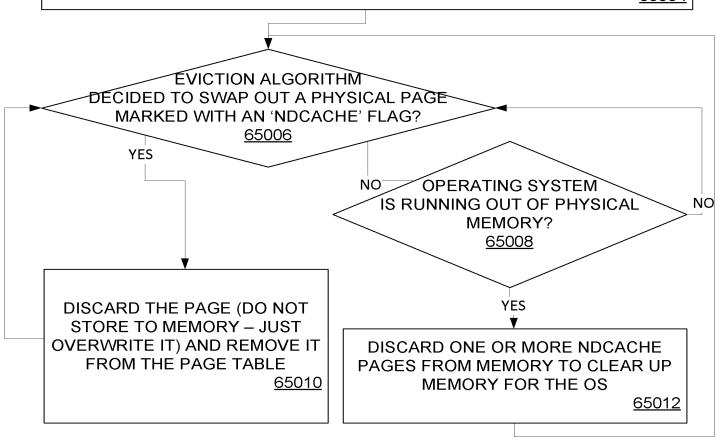
LOAD THE DATA IN TO THE PHYSICAL MEMORY OR MAP IT 64816

#### FIG. 650 - NDCACHE FLOWCHART

APPLICATION REQUESTS FROM THE OPERATING SYSTEM A NON-DETERMINISTIC PHYSICAL MEMORY CACHE (NDCACHE)

65002

THE OS ALLOCATES THE PHYSICAL MEMORY FOR THE CACHE, MAPS IT TO VIRTUAL MEMORY, AND MARKS THESE PAGES WITH AN 'NDCACHE' FLAG 65004



WHEN READING A SWAPPED OUT NDCACHE PAGE, GET BACK A 'DOES\_NOT\_EXIST' 65014 DURING A READ/WRITE TRANSACTION TO/ FROM AN NDCACHE PAGE, LOCK THAT PAGE (I.E. DO NOT LET THE EVICTION ALGORITHM SWAP OUT THAT PAGE DURING THE READ/WRITE TRANSACTION) 65016

#### FIG. 654 - NDCACHE – API

#### API (SIMILAR TO THE FILE API IN POSIX)

#### FILEDS NDCACHE OPEN("NAME", CREATE FLAG, TRUNCATE FLAG)

- DESCRIPTION: OPEN AN NDCACHE CALLED 'NAME' FOR READING OR WRITING AND RECEIVE A HANDLE [FILEDS] 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 (FILEDS) TO BE USED AS A HANDLE FOR THE OTHER NDCACHE APIS

65402

## NDCACHE READ(FILEDS, BUFFER, SIZE) NDCACHE WRITE(FILEDS, BUFFER, SIZE)

- DESCRIPTION: READ OR WRITE TO THE NDCACHE REFERENCED BY FILEDS
- FILEDS 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(FILEDS)

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

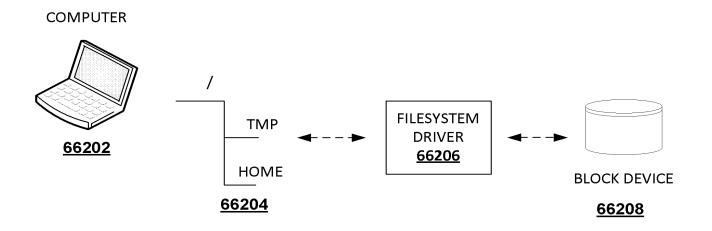
65406

#### NDCACHE UNLINK(FILEDS)

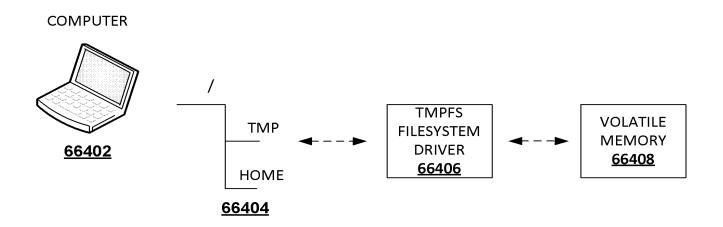
- DESCRIPTION: REMOVE AN NDCACHE ENTRY FROM THE PAGE TABLE
- FILEDS THE FILE DESCRIPTOR OF THE OPEN NDCACHE TO REMOVE FROM THE PAGE TABLE

65408

## FIG. 662 – BACKGROUND - MOUNTING A FILESYSTEM



# FIG. 664 - BACKGROUND - TMPFS FILE SYSTEM



# FIG. 666 – NDCACHE – SECONDARY IMPLEMENTATION METHOD USING FS MOUNTING - IMPLEMENTATION

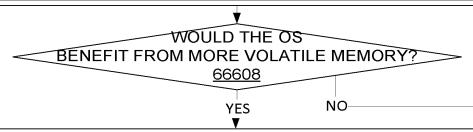
CREATE AND MOUNT A NEW TMPFS WITH AN INITIAL SIZE OF X BYTES (COULD BE 1GB FOR EXAMPLE) TO BE USED FOR ND CACHE 66602

PERIODICALLY CHECK WHETHER THE OS WOULD BENEFIT FROM HAVING ADDITIONAL VOLATILE MEMORY 66604

IT IS POSSIBLE TO CHECK WHETHER OS NEEDS MORE MEMORY BY ONE OR MORE OF THE FOLLOWING METHODS:

- 1. PERIODIC FREE MEMORY CHECK EVERY X MS (COULD BE EVERY 20MS) FLAG IF:
- A. LESS THAN A CERTAIN AMOUNT OF FREE MEMORY EXISTS IN THE SYSTEM (COULD BE LESS THAN 200MB)
- B. ABOVE A NUMBER OF MEMORY SWAPS OCCUR IN THE OS (COULD BE MORE THAN 1 SWAP PER 500MS)
- C. KERNEL CACHE SPACE REDUCED TO UNDER A CERTAIN SIZE (COULD BE UNDER 2GB)
- 2. LISTEN TO 'STRESS' OS CALLS, AND FLAG IF ANY OF THE STRESS CALLS HAVE BEEN INVOKED. THESE COULD BE FOR EXAMPLE "LOW MEMORY" TYPE CALL BACKS
- 3. ATTACH CALLBACK ON SWAPPING OUT OF THE TMPFS FILE, AND DELETING IT INSTEAD OF SWAPPING OUT

66606



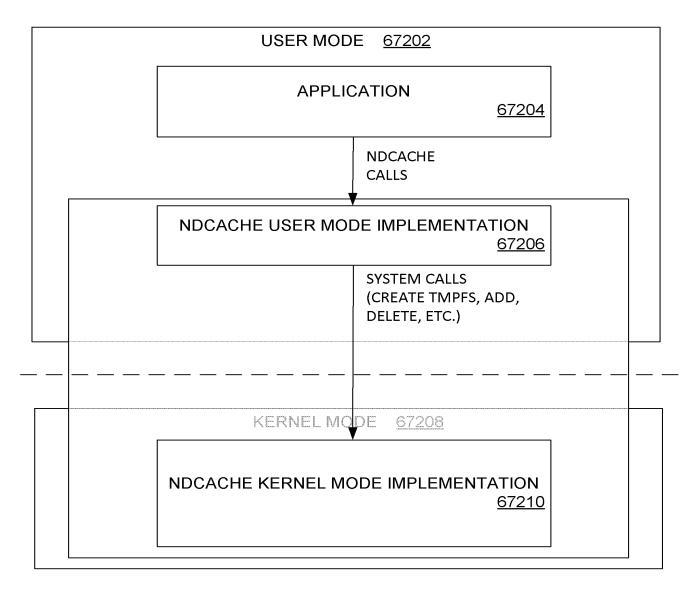
DELETE A NUMBER OF ELEMENTS FROM THE NDCACHE SO THAT A SET CHUNK OF IT WILL BE FREE (CHUNK COULD BE 200MB FOR EXAMPLE) 66610

IT IS POSSIBLE TO DELETE ELEMENTS FROM THE NDCACHE (TMPFS) BY USING ONE OR MORE OF THE FOLLOWING METHODS:

- 1. LRU DELETE THE LEAST RECENTLY USED (IN TERMS OF READ/WRITE) ELEMENTS FROM THE TMPFS
- 2. LFU DELETE THE LEAST FREQUENTLY USED (IN TERMS OF READ/WRITE) ELEMENTS FROM THE TMPFS
- 3. SIZE DELETE THE LARGEST ELEMENTS (BYTES) FROM THE TMPFS
- 4. SIZE DELETE THE SMALLEST ELEMENTS (BYTES) FROM THE TMPFS

NetNut Ltd. v. Brig 666th 2td. IPR2021-00465, EX. 2023 Page 108 of 121

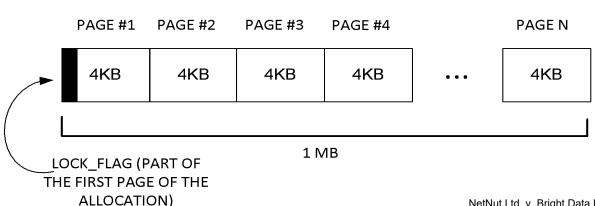
# FIG. 672 – NDCACHE – THIRD IMPLEMENTATION METHOD - USER & KERNEL MODE



# FIG. 674 – NDCACHE – THIRD IMPLEMENTATION METHOD – HIGH LEVEL IMPLEMENTATION

WRITING TO (AND ALLOCATING) THE NDCACHE:

## FIG. 675 – NDCACHE PAGES AND LOCK FLAG



# FIG. 676 – NDCACHE – THIRD IMPLEMENTATION METHOD– HIGH LEVEL IMPLEMENTATION (CONT.)

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
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;
                                                                67602
```

# FIG. 678 – NDCACHE – THIRD IMPLEMENTATION METHOD – HIGH LEVEL IMPLEMENTATION (CONT.)

NDCACHE KERNEL MODULE (NDCACHE DRIVER):

**EVICTION 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

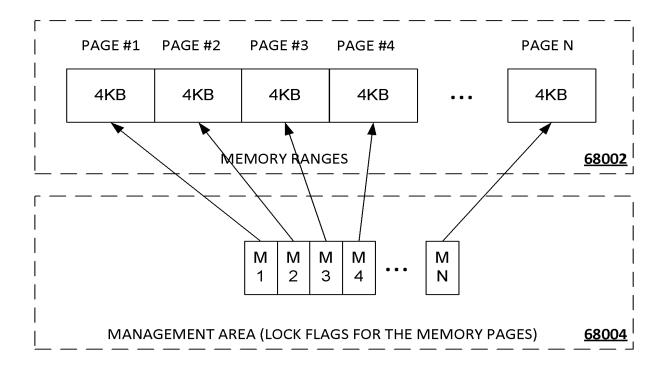
67802

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)

## FIG. 680 – NDCACHE – FOURTH METHOD – MULTIPLE SEGMENTS IN CACHE

#### FOURTH METHOD - ONE MANAGEMENT AREA PER PAGE:



#### FIG. 682 – NDCACHE – FOURTH METHOD – FLOW CHART

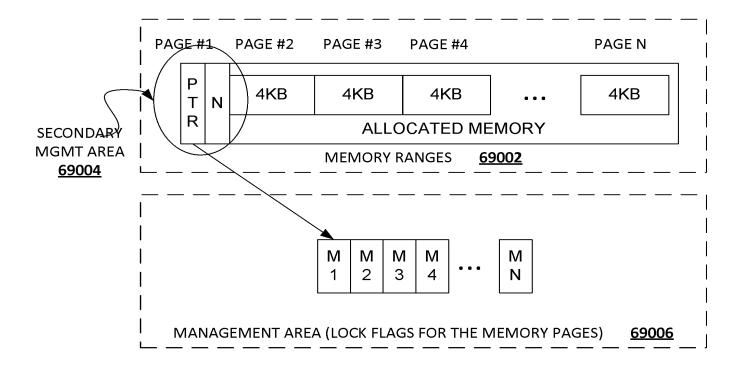
**INITIALIZATION:** 

ALLOCATE MEMORY FOR THE NDCACHE FROM THE OS (E.G. ALLOCATE 20MB)

<u>68202</u>

ALLOCATE A MANAGEMENT AREA, WHOSE SIZE IS [MEMORY\_ALLOCATED / SIZEOF(MEM\_PAGE)\*SIZEOF(INT) ].

### FIG. 690 – NDCACHE – IMPROVEMENT ON THE FOURTH METHOD – MEMORY LAYOUT



#### FIG. 692 – NDCACHE – IMPROVEMENT ON THE FOURTH METHOD – FLOWCHART

**NEW NDCACHE ELEMENT ALLOCATION: P = NDCACHE\_CREATE(SIZE):** 

ALLOCATE MEMORY (P) FOR THE NDCACHE ACCORDING TO THE SIZE REQUESTED BY THE CALLER (SIZE) + SIZEOF(T) WHERE T IS THE SPACE REQUIRED FOR THE SECONDARY MANAGEMENT AREA (SIZEOF(POINTER) + SIZEOF(INT))

69202

ALLOCATE A MANAGEMENT AREA FOR THE PAGES' 'LOCK FLAGS' (THE SIZE OF IT IS ONE INTEGER \* NUMBER OF PAGES REQUIRED FOR THE NDCACHE)

69204

DEFINE A MANAGEMENT AREA (T) AT THE START OF P, AND INSERT INTO IT A POINTER TO THE MANAGEMENT AREA (T-PTR) AND AN INTEGER (T->N) THAT REPRESENTS THE NUMBER OF PAGES IN THIS NDCACHE'S MEMORY RANGE

69206

P=P+SIZEOF(T) – SO THAT P NOW POINTS TO THE MAIN MEMORY RANGE (RIGHT AFTER THE SECONDARY MANAGEMENT AREA) 69208

RETURN (P+SIZEOF(T)) - (RETURNS A POINTER TO THE PLACE IN THE MEMORY WHERE THE NDCACHE MEMORY ITSELF STARTS (RIGHT AFTER THE SECONDARY MANAGEMENT SECTION)

### FIG. 696 – NDCACHE – IMPROVEMENT ON THE FOURTH METHOD – FLOWCHART (CONT.)

#### 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

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

#### FIG. 700 – IDLE MONITOR

APPLICATION THAT NEEDS TO BE NOTIFIED OF 'IDLENESS' NOTIFIES OF ITS REQUEST, PROVIDES A CALLBACK (CB) AND ON WHICH IDLENESS PARAMETERS IT WISHES TO GET CALLED ON (PARAMS), AND THE DURATION OF THE IDLENESS REQUIRED TO CALL CB (T)

70002

70004

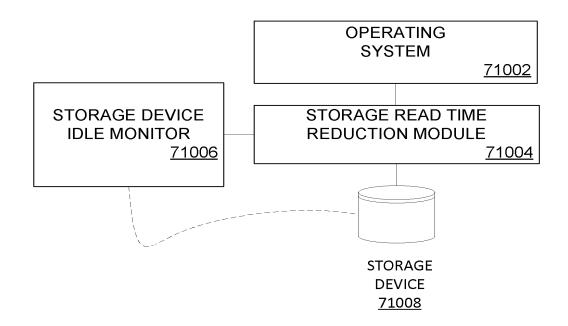
#### SCAN IDLENESS SOURCES. CAN BE ANY OF:

- BITS BEING SENT OR RECEIVED TO/FROM COMMUNICATION DEVICE
- HARD DISK READ OR WRITE
- TEMPERATURE CHANGES IN ANY OF THE DEVICE'S TEMPERATURE GAUGES
- CPU BUSY OVER CERTAIN THRESHOLD (COULD BE 5% FOR EXAMPLE)
- CAMERA SENSES MOTION
- LIGHT SENSOR SENSES CHANGES IN LIGHT INTENSITY OR STRUCTURE
- ACCELEROMETER THAT SENSES MOVEMENT OR ORIENTATION CHANGE
- HD ACCELEROMETER THAT SENSES MOVEMENT
- GPS SENSOR THAT SENSES MOVEMENT

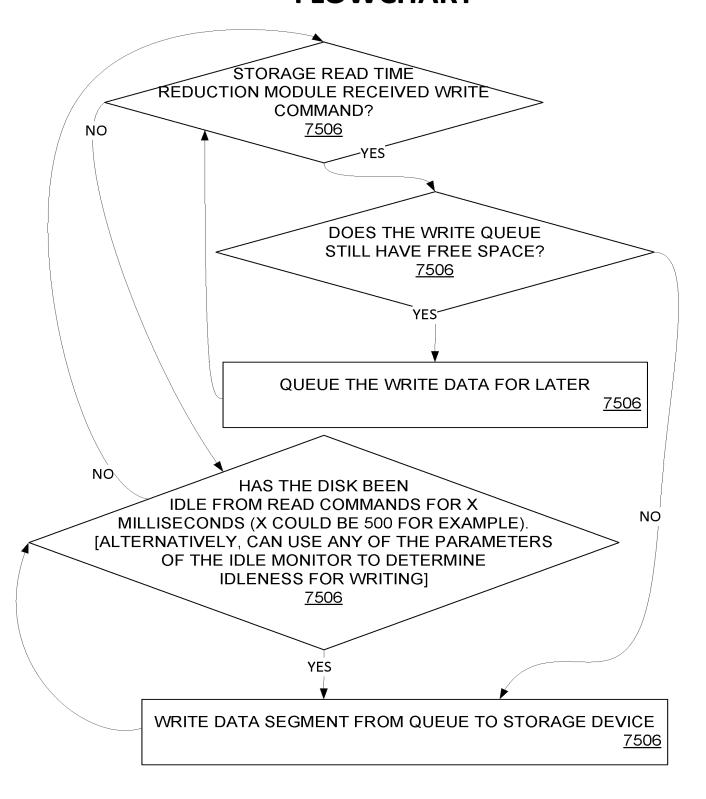
IDLE
RESPONSE RECEIVED FROM
ANY SENSOR?
70006

CALL THE APPLICATION'S CB IF THE SENSOR THAT ARE IDLE ARE WHAT THE APPLICATION REQUESTED TO BE NOTIFIED ABOUT IN PARAMS, AND IF THEY'VE ALL BEEN IDLE FOR T SECONDS

### FIG. 710 – STORAGE WRITE ON IDLE; REDUCING STORAGE READ TIMES -DIAGRAM



### FIG. 712 – STORAGE WRITE ON IDLE; REDUCING STORAGE READ TIMES -FLOWCHART



## FIG. 420 – AP SELECT ALGORITHM - FLOWCHART

CREATE LIST OF ACCESS POINTS (AP\_LIST) THAT ARE SEEN BY THE CLIENT'S WIFI DRIVER. FOR EACH AP, STORE THE FOLLOWING INFORMATION:

SIGNAL STRENGTH, AUTHENTICATION TYPE

42002

ADD TO EACH OF THE ACCESS POINTS IN THE AP\_LIST THE FOLLOWING INFORMATION, BASED ON THE DATABASE OF PREVIOUS CONNECTIONS THAT THIS CLIENT HAS WITH THE ACCESS POINTS AROUND IT:

HAS CLIENT EVER TRIED TO CONNECT TO THIS AP?, HAS THIS CLIENT SUCCEEDED IN CONNECTING TO THIS AP IN THE PAST? WAS THE INTERNET CONNECTION VERIFIED? TO WHICH AP WAS THIS CLIENT CONNECT TO LAST?, WHAT WAS THE LAST REASON FOR DISCONNECT?

42004

TRY TO CONNECT TO THE APS IN THE AP\_LIST ACCORDING TO THE FOLLOWING ORDER (WHERE THERE ARE SEVERAL CANDIDATES FOR EACH STAGE, SORT THEM BY SIGNAL STRENGTH):

- LOCKED ACCESS POINTS THAT THIS HOST VERIFIED CONNECTION TO INTERNET WITH TO IN THE PAST
- OPEN ACCESS POINTS THAT THIS HOST VERIFIED CONNECTION TO INTERNET WITH IN THE PAST
- OPEN ACCESS POINTS THAT THIS HOST NEVER VERIFIED WITH IN THE PAST
- OPTIONAL: LOCKED APS THAT WERE NEVER VERIFIED WITH BY THIS HOST, WITH GUESSED PASSWORD (SEE OTHER SECTION ON HOW TO GUESS)
- THE REST OF THE ACCESS POINTS IN THE LIST

<u>42006</u>

ONCE CONNECTED AND VERIFIED, ADD INFORMATION ABOUT THIS ACCESS POINT AND ABOUT THE CONNECTION TO THE DATABASE OF PREVIOUS CONNECTIONS ON THIS HOST