

(51) International Patent Classification:
H04L 29/08 (2006.01)(21) International Application Number:
PCT/US2015/020780(22) International Filing Date:
16 March 2015 (16.03.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/969,711	24 March 2014 (24.03.2014)	US
61/969,708	24 March 2014 (24.03.2014)	US
14/554,711	26 November 2014 (26.11.2014)	US
14/554,739	26 November 2014 (26.11.2014)	US

(71) Applicant: **TANIUM INC.** [US/US]; 2200 Powell Street
6th Floor, Emeryville, CA 94608 (US).

(72) Inventors: **HINDAWI, David**; 2200 Powell Street, 6th
Floor, Emeryville, CA 94608 (US). **HINDAWI, Orion**;
2200 Powell Street, 6th Floor, Emeryville, CA 94608 (US).
LIPPINCOTT, Lisa; 2200 Powell Street, 6th Floor,
Emeryville, CA 94608 (US). **LINCROFT, Peter**; 2200
Powell Street, 6th Floor, Emeryville, CA 94608 (US).

(74) Agents: **WILLIAMS, Gary, S.** et al.; Morgan Lewis &
Bockius LLP, 2 Palo Alto Square, 3000 El Camino Real,
Suite 700, Palo Alto, CA 94306 (US).

(81) Designated States (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY,
BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM,
DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,
HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR,
KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG,
MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM,
PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC,
SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ,
TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU,
TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,
DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU,
LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,
SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,
GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

(54) Title: LOCAL DATA CACHING FOR DATA TRANSFERS ON A NETWORK OF COMPUTATIONAL DEVICES AND
DATA CACHING AND DISTRIBUTION IN A LOCAL NETWORK

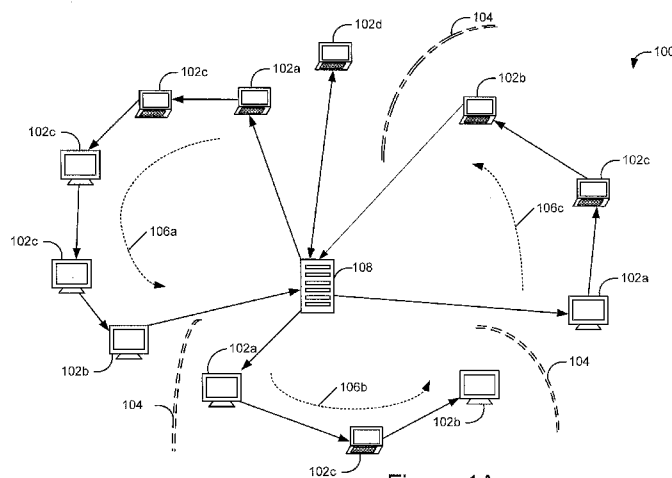


Figure 1A

(57) Abstract: In accordance with some embodiments, a computational machine having one or more processors, a local cache and memory receives from a first machine a data request that is used to request specific data. The computational machine determines whether the computational machine stores the specific data in the local cache. In accordance with a determination that the computational machine does not store the specific data in the local cache, the computational machine responds to the data request by passing the data request to a second machine and returning the specific data to the first machine when the second machine returns the specific data to the computational machine. Additionally, the computational machine determines whether to store the returned specific data in the local cache according to a data caching method.



Local Data Caching for Data Transfers on a Network of Computational Devices and Data Caching and Distribution in a Local Network

BACKGROUND

[0001] There are many circumstances in which multiple computational machines (sometimes called computers) on the same local area network (LAN) need the same specific data file (e.g., a printer driver, a dll, etc.), the same software application, or the same update to a software application. Data transfers currently make heavy use of both hardware (distributed file servers) and wide area network (WAN) bandwidth, for example, by having each computation machine on the local area network separately obtain a copy of the same application or update from a resource external to the local area network. It would be beneficial to have a more efficient data transfer mechanism than the current practice.

SUMMARY

[0002] In one aspect, a method of data caching and distribution includes, at a computational machine having a local cache, receiving from a first machine a data request that is used to request specific data. The method further includes determining whether the computational machine stores the specific data in the local cache, and in accordance with a determination that the computational machine does not store the specific data in the local cache, responding to the data request, by: passing the data request to a second machine, and when the second machine returns the specific data to the computational machine, returning the specific data to the first machine and determining whether to store the specific data in the local cache according to a data caching method.

[0003] In some embodiments, the method further includes, in accordance with a determination that the computational machine stores the specific data in its local cache, responding to the data request by returning the specific data to the first machine.

[0004] In some embodiments, the data caching method comprises a statistically random method of determining whether to cache respective data received from another machine.

[0005] In some embodiments, the data caching method is based on at least one of the size of the specific data, the size of the local cache, the size of available caches in a linear communication orbit that includes the computational machine, and size of a respective data set that includes the specific data.

[0006] In some embodiments, the specific data comprises one of a plurality of shards of an object, such as a data file, software application, software application update, software patch, data file, database, or database update.

[0007] In some embodiments, the computational machine is a head node of a linear communication orbit that is organized from a plurality of machines, including the computational machine and the first machine, based on a set of rules at a specific time, and the second machine is a server coupled to the head node.

[0008] In some embodiments, the data request comprises a first data request, and the method includes receiving at the computational machine, after receiving the first data request, a second data request to request the same specific data as the first data request, and in accordance with a determination that predefined criteria are satisfied, forgoing passing the second data request to the second machine.

[0009] In some embodiments, the computational machine is an intermediate machine between neighboring machines within a linear communication orbit, the linear communication orbit being organized from a plurality of machines based on a set of rules at a specific time and subject to changes in configuration and membership when one or more machines join or exit the linear communication orbit.

[0010] In some embodiments, the computational machine is downstream of the first machine and upstream of the second machine in the linear communication orbit. Alternatively, in some embodiments, the computational machine is downstream of the second machine and upstream of the first machine in the linear communication orbit.

[0011] In some embodiments, the computational machine and the first machine are interconnected by a single local area network.

[0012] In another aspect, another method of data caching and distribution includes, at a computational machine having a local cache, generating a first data request that is used to request specific data. The method further includes passing the data request to a second

machine; receiving at the computational machine, from a third machine, a second data request to request the same specific data after generating the first data request; in accordance with a determination that predefined criteria are satisfied, forgoing passing the second data request to the second machine; and when the second machine returns the specific data to the computational machine, returning the specific data to the third machine.

[0013] In another aspect of the invention, a non-transitory computer-readable medium, having instructions stored thereon, which when executed by one or more processors cause the processors of a computation machine, having a local cache, to receive from a first machine a data request that is used to request specific data, determine whether the computational machine stores the specific data in the local cache, and in accordance with a determination that the computational machine does not store the specific data in the local cache, respond to the data request, by: passing the data request to a second machine, and when the second machine returns the specific data to the computational machine, returning the specific data to the first machine and determining whether to store the specific data in the local cache according to a data caching method.

[0014] In some embodiments, the instructions stored in the non-transitory computer-readable medium include instructions for causing the computational machine to perform the method of any of the methods described above.

[0015] In another aspect of the invention, a computational machine includes one or more processors, a local cache, and memory having instructions stored thereon, which when executed by the one or more processors cause the computational machine to receive from a first machine a data request that is used to request specific data, determine whether the computational machine stores the specific data in the local cache, and in accordance with a determination that the computational machine does not store the specific data in the local cache, respond to the data request, by: passing the data request to a second machine, and when the second machine returns the specific data to the computational machine, returning the specific data to the first machine and determining whether to store the specific data in the local cache according to a data caching method.

[0016] In some embodiments, the instructions stored in the memory include instructions for causing the computational machine to perform the method of any of the methods described above.

[0017] In yet another aspect, a method of data caching and distribution, performed by a plurality of machines in a linear communication orbit, includes generating a data request by a first machine to request specific data, and passing the data request along a data request path that tracks the linear communication orbit until the request is received at a second machine, in the linear communication orbit, that returns the specific data in response to the data request. The method includes, at a third machine between the second machine and the first machine in the linear communication orbit, conditionally storing the specific data in a local cache of the third machine according to a data caching method.

[0018] In some embodiments, the second machine is a machine, in the linear communication orbit, that has a cached copy of the specific data in its local cache or the head node of the linear communication orbit if no machine in the request path has a cached copy of the specific data in its local cache.

[0019] In some embodiments, the second machine is a machine closest, along the data request path, to the first machine that has a cached copy of the specific data in its local cache.

[0020] In some embodiments, passing the data request includes passing the data request along the data request path that tracks the linear communication orbit until (A) the request is received at a machine, in the linear communication orbit, that has a cached copy of the specific data in a local cache of the machine or (B) the request is received at a head node of the linear communication orbit, which obtains and returns the specific data in response to the request.

[0021] In some embodiments, the plurality of machines that are included in the linear communication orbit are interconnected by a single local area network.

[0022] In some embodiments, the data caching method comprises a statistically random method of determining whether to cache respective data received from another machine.

[0023] In some embodiments, the specific data comprises one of a plurality of shards, in an object, to be cached in the linear communication orbit, and the data caching method is based on at least one of the size of the specific data, the size of the object, the size of the local cache of the third machine, a total size of available caches in the linear communication orbit, and a total size of a data set that includes the object.

[0024] In some embodiments, the data request is a first data request and the specific data is first specific data, and the method includes generating, at the first machine, a second data request to request second specific data, passing the second data request along the data request path until the request is received at a fourth machine, in the linear communication orbit, that returns the second specific data in response to the second data request, wherein the fourth machine is distinct from the second machine. In these embodiments, the method further includes combining, at the first machine, a plurality of specific data, including the first specific data and the second specific data, to generate an object for use at the first machine.

[0025] In some embodiments, the plurality of specific data comprise shards of the object specified in a manifest.

[0026] In some embodiments, generating the first data request by the first machine includes delaying generation of the first data request, with respect to time at which the manifest is received by the first machine, in accordance with a predefined request spreading method.

[0027] In some embodiments, the data request is a first data request, and the method further includes generating at a fifth machine a third data request to request the specific data after generating the first data request by the first machine, and in accordance with a determination that predefined criteria in a request consolidation method are satisfied, forgoing passing the third data request along at least a part of a respective data request path that is associated with the third data request.

[0028] In some embodiments, the specific data are conveyed to the fifth machine by returning the specific data to the first machine, which reconveys the specific data to the fifth machine.

[0029] In some embodiments, the specific data is one of a plurality of shards of an object, such as a software application, software application update, database, or database update.

[0030] In some embodiments, the method includes passing the specific data from the second machine to the first machine through a plurality of intervening machines in the linear communication orbit, and, at each machine of the plurality of intervening machines, located between the second machine and the first machine, caching a copy of the specific data in a

local cache of the machine in accordance with a determination that predefined local caching criteria are met.

[0031] In some embodiments, the method includes passing the specific data from the second machine to the first machine through a plurality of intervening machines in the linear communication orbit and continuing to pass the specific data to additional machines in the linear communication orbit until the specific data are received by one of the end nodes. In these embodiments, method includes, at each machine of a plurality of machines, selected from the plurality of intervening machines and the additional machines, caching a copy of the specific data in a local cache of the machine in accordance with a determination that predefined local caching criteria are met.

[0032] In some embodiments, the second machine is a downstream machine between the first machine and the tail node of the linear communication orbit, the data request being passed downstream from the first machine to the second machine, the specific data being returned upstream from the second machine to the first machine.

[0033] In some embodiments, the second machine is an upstream machine between the first machine and the head node of the linear communication orbit, the data request being passed from the first machine downstream to the tail node and upstream to the second machine, the specific data being returned downstream from the second machine to the first machine.

[0034] In some embodiments, the second machine is the tail node or the head node of the linear communication orbit, the specific data being returned from the second machine to the first machine and conditionally stored in the plurality of machines, including the third machine, that are coupled within the linear communication orbit and do not have the specific data.

[0035] Other embodiments and advantages may be apparent to those skilled in the art in light of the descriptions and drawings in this specification.

BRIEF DESCRIPTION OF THE DRAWINGS

[0036] Figure 1A illustrates a managed network comprising a plurality of interconnected machines, e.g., computers, servers, mobile devices, and other networked devices.

[0037] Figure 1B-1D illustrate respective local segments of an exemplary linear communication orbit for data caching and distribution in a managed network in accordance with some embodiments.

[0038] Figure 2A illustrates an exemplary data caching and distribution system that processes a system management message for file transfers, information installations or updates in accordance with some embodiments.

[0039] Figure 2B illustrates an exemplary data caching and distribution system that contains a data request path for propagating a data request and requesting specific data in a linear communication orbit in accordance with some embodiments.

[0040] Figures 3A-3C illustrate three respective linear communication orbits that are associated with three scenarios of transferring a data request and distributing specific data based on a location of the specific data in accordance with some embodiments.

[0041] Figure 4 illustrates an exemplary process of providing the data requested by two respective data requests, both generated by a computational machine on a linear communication orbit in accordance with some embodiments.

[0042] Figure 5 illustrates an exemplary process of consolidating data requests issued by two distinct computational machines to request the same specific data on a linear communication orbit in accordance with some embodiments.

[0043] Figure 6A is a block diagram of an exemplary computational machine in the exemplary linear communication orbits shown in Figures 1A-5 in accordance with some embodiments.

[0044] Figure 6B is a block diagram of an exemplary server in the exemplary linear communication orbits shown in Figure 1A-5 in accordance with some embodiments.

[0045] Figure 7 illustrates an exemplary computational machine that responds to an incoming data distribution command in accordance with some embodiments.

[0046] Figure 8A illustrates an exemplary intermediate machine that processes a data request and conditionally stores specific data requested by the data request in accordance with some embodiments.

[0047] Figure 8B illustrates an exemplary computational machine that acts as a head node of a linear communication orbit and conditionally stores specific data during the course of processing a corresponding data request in accordance with some embodiments.

[0048] Figure 9 illustrates an exemplary computational machine that consolidates two data requests that request the same specific data in accordance with some embodiments.

[0049] Figure 10A illustrates a conceptual block diagram of a linear communication orbit that distributes and caches specific data in accordance with some embodiments.

[0050] Figure 10B illustrates a conceptual block diagram of another linear communication orbit that distributes and caches specific data in accordance with some embodiments.

[0051] Figures 11A-11D illustrate a flow diagram representing a method for caching and distribution specific data in accordance with some embodiments.

[0052] Figures 12A-12G illustrate a flow diagram representing another method for caching and distributing specific data in accordance with some embodiments.

[0053] Like reference numerals refer to corresponding parts throughout the drawings.

DESCRIPTION OF EMBODIMENTS

[0054] Figure 1A illustrates a managed network 100 comprising a plurality of interconnected machines 102 (including 102a-d), e.g., computers, servers, mobile devices, and other networked devices. Examples of the managed network 100 include an enterprise network or another network under common management. The managed network 100 a plurality of sub-networks (e.g., contiguous segments 106), and optionally includes one or more singleton machine (e.g., singleton 102d). Each singleton or sub-network is coupled to a server 108 that facilitates creation, maintenance and operation of the respective singleton and sub-network. This server 108 may be elected automatically from among all or a subset of machines 102 according to various predetermined election rules implemented on machines 102. Each sub-network in the managed network 100 further includes at least two interconnected machines and adopts a certain network topology to organize these machines. Each singleton machine, if any, is a stand-alone node except that it is coupled to server 108.

In some embodiments, each singleton or sub-network is further separated from the rest of the managed network 100 by one or more firewalls 104.

[0055] Optionally, machines 102 in managed network 100 are distributed across different geographical areas. Alternatively, machines 102 are located at the same physical location. A respective machine 102 communicates with another machine 102 or the server 108 using one or more communication networks. Such communications include communications during normal operations (e.g., user-level operations, such as emailing, Internet browsing, VoIP, database accessing, etc.). The communication network(s) used can be one or more networks having one or more type of topologies, including but not limited to the Internet, intranets, local area networks (LANs), cellular networks, Ethernet, Storage Area Networks (SANs), telephone networks, Bluetooth personal area networks (PAN) and the like. In some embodiments, two or more machines 102 in a sub-network are coupled via a wired connection, while at least some of machines in the same sub-network are coupled via a Bluetooth PAN.

[0056] Machines 102 in managed network 100 are organized into one or more contiguous segments 106 (including 106a-c), each of which becomes a sub-network in the managed network 100. In some embodiments, each contiguous segment 106 is a respective linear communication orbit that supports system, security and network management communications within the managed network 100. Furthermore, each contiguous segment 106 includes one head node (e.g., head node 102a), one tail node (e.g., tail node 102b), and a sequence of zero or more intermediate client nodes (e.g., intermediate node(s) 102c) in between the head node and the tail node. In some embodiments, both the head node and tail node of a contiguous segment 106a are coupled to server 108, while the intermediate nodes of the contiguous segment 106a are not coupled to server 108. In some embodiments, only the head node of a contiguous segment 106b is coupled to the server 108, while the intermediate nodes and tail node are not coupled to the server 108.

[0057] In some embodiments, all machines 102 coupled to a linear communication orbit 106 in network 100 are sorted into an ordered sequence according to a respective unique identifier associated with each machine 102. For example, respective IP addresses of machines 102 are used to sort the machines into an ordered sequence in the linear communication orbit. Each machine is provided with a predetermined set of rules for

identifying its own predecessor and/or successor nodes given the unique identifiers of its potential neighbor machines. When a machine joins or leaves the linear communication orbit, it determines its ordinal position relative to one or more other machines in the linear communication orbit according to the unique identifiers and the aforementioned rules. More details on how a linear communication orbit is organized and how each intermediate node, head node or end node enters and leaves the linear communication orbit are provided in the Applicants' prior application, U.S. Patent Application Serial No. 13/797,962, filed March 12, 2013, entitled "Creation and Maintenance of Self-Organizing Communication Orbits in Distributed Networks," which is hereby incorporated by reference in its entirety.

[0058] Linear communication orbits, such as exemplary linear communication orbits 106a-106c, are established and maintained to facilitate system, security and/or network management operations ascribed to manual and programmed administration of network 100. Examples of system, security and network management operations include: (1) collecting status information (e.g., bandwidth, load, availability, resource inventory, application status, machine type, date of last update, security breach, errors, etc.) from individual machines of the managed network; (2) issuance of system, security and network management commands (e.g., commands related to shutdown, restart, failover, release of resources, change access authorizations, backup, deployment, quarantine, load balancing, etc.) for individual resources and/or machines on the managed network; (3) file distribution, including software installations and updates; (4) detecting presence of particular malicious programs (e.g., viruses, malware, security holes, etc.) on individual machines on the managed network; (5) removal of or disabling particular malicious programs (e.g., viruses, malware, security holes, etc.) on individual machines on the managed network; (6) disabling or suspending suspicious or high-risk operations and activities (e.g., Internet or operating system activities of suspected virus, malware, etc.) on particular machines on the managed network; (7) detecting unmanaged machines coupled to the managed network; (8) detecting data leakage (e.g., transmission of classified information) from machines on the managed network to locations or machines outside of the managed network; (9) detecting connection or data transfer to/from removable data storage devices (e.g., memory stick, or other removable storage devices) from/to particular ports (e.g., a USB drive) of particular machines on the managed network. Other system, security and network management operations are possible, as will be apparent to those of ordinary skills in the art.

[0059] The present specification focuses on data caching and data distribution among the computational machines 102 located at different client nodes of an established linear communication orbit. Such data caching and data distribution operations include, for example, enterprise software installations and updates, software patch installations, anti-virus updates, obtaining a local copy of data file, database updates, etc., within the linear communication orbits 106a-106c.

[0060] In some implementations, a system management message is initially issued by the server (e.g., server 108) in the managed network, and includes a command (e.g., a command to obtain and install a specific object, such as a data file or software application, or software update) and a rule. In some embodiments, the system management message is initially received at a head node of an established linear communication orbit, as is then passed along the linear communication orbit to each node in the linear communication orbit until it reaches the tail node of the linear communication orbit. Each node of the linear communication orbit is sometimes called a computational machine; alternatively, each node of the linear communication orbit includes a computational machine.

[0061] The rule in the system management message is interpreted by each computational machine in the orbit, which determines whether that machine should execute the command. Alternately stated, the rule, when interpreted by each computational machine in the orbit, determines whether that machine needs a specific object, such as a data file, an application, an update, or the like. If the particular machine 102 determines that it satisfies the rule, and thus needs the object, it generates a plurality of data requests to request a plurality of shards, as described in more detail below. Each of the data requests is a request for respective specific data, herein called a shard. Together the shards form, or can be combined to form, an object, such as a file, an application, or an update.

[0062] Each data request propagates along a predetermined data request path that tracks the linear communication orbit until the requested respective specific data are identified at a respective machine that stores a cached copy of the requested respective specific. The respective specific data are thereafter returned, along a data return path, to the computational machine 102 that made the request. Moreover, the requested specific data are selectively cached at the other computational machines located on the path to facilitate potential future installations and updates at other machines on the linear communication

orbit. During the entire course of data caching and distribution, each individual computational machine on the linear communication orbit follows a predetermined routine to independently process system management messages, respond to any incoming data request and cache specific data that passes through.

[0063] Data distribution in the linear communication orbit(s) 106 shown in Figure 1A is preferably controlled locally within the linear communication orbit, and many data management functions are distributed to each individual machine. For example, machines located on a data return path independently determine whether to retain a cached copy of data being returned to another machine on the linear communication orbit. Further, each machine located on a data request path independently determines if it has a cached copy of the specific data requested by another machine on the linear communication orbit, and if so, returns the specific data requested from its local cache. The server 108 need not separately manage these functions. Machines located on the data request path determine successively and independently whether each of them retains a cached copy and thereby should return the specific data.

[0064] In accordance with many embodiments of the present invention, data caching and distribution is performed using local resources of a linear communication orbit, except when none of the machines on the linear communication orbit have the specific data requested by a machine on the linear communication orbit. Bandwidth on the local area network, which interconnects the machines on the linear communication orbit, is typically very low cost compared with bandwidth on the wide area network that connects a server to one or more machines on the linear communication orbit. In many implementations, the local area network also provides higher data communication rates than the wide area network. Distributed, automatic local caching minimizes or reduces the use of a wide area communication network to retrieve data from a server, particularly when multiple machines on the same linear communication orbit make requests for the same data. As a result, overall performance of a large-scale managed network is improved compared with systems in which each managed machine obtains requested data via requests directed to a server via a wide area network. Due to local caching of content, and locally made caching decisions, far fewer file servers are needed in systems implementing the methods described herein, compared with conventional, large-scale managed networks of computational machines.

[0065] Figure 1B-1D illustrate respective local segments of an exemplary linear communication orbit for data caching and distribution in a managed network in accordance with some embodiments. Figure 1B illustrates a local segment 112 that is centered about an intermediate client node 114. Intermediate node 114 is coupled between two immediately adjacent nodes, i.e., predecessor node 124 and successor node 126. Each machine 102 other than a head node and a tail node in a linear communication orbit is such an intermediate node. In some embodiments, an intermediate node is persistently connected only to its predecessor and successor nodes, and not to the server or other nodes in the network.

[0066] Figure 1C illustrates a local segment (e.g., local segment 116) that includes a head node 118 of the linear communication orbit. Figure 1D illustrates a local segment 120 that includes a tail node 122 of the linear communication orbit in accordance with some embodiments. Head node 118 and tail node 122 are each coupled to an immediately adjacent node, successor node 128 and predecessor node 130, respectively, on the linear communication orbit. In addition, head node 118 is coupled to server 132 that serves as a pseudo-predecessor node for head node 118. In some embodiments, tail node 122 is also coupled to server 132 that serves as a pseudo-successor node for tail node 122; however, in some embodiments, the linear communication orbit terminates at tail node 122. Each node shown in Figures 1B-1D is implemented by a respective machine 102 coupled to one of the linear communication orbits (e.g., linear communication orbits 106a-106c) of network 100 in Figure 1A.

[0067] Each machine on the linear communication orbit is coupled to an immediate neighbor machine or a server via a bidirectional communication link that includes a data incoming channel and a data outgoing channel. Intermediate machine 114 has four communication channels (e.g., channels 134-140) that form two bidirectional communication links to couple itself to two respective neighbor machines (e.g., predecessor node 124 and successor node 126). Head node 118 and tail node 122 are similarly coupled to a respective neighbor machine, and in some embodiments to a server, via bidirectional communication links. The bidirectional communication links allow each machine to simultaneously receive information from and provide information to its adjacent machine upstream or downstream in the linear communication orbit. The upstream direction, relative to a machine other than the head node, is the direction of communication or message flow toward the head node. For the

head node, the upstream direction is toward the server to which the head node is coupled. The downstream direction, relative to a machine other than the head node, is the direction of communication or message flow toward the tail node. In some embodiments, for the tail node, the downstream direction is undefined. In some other embodiments, for the tail node, the downstream direction is toward the server to which the tail node is coupled.

[0068] As shown in Figure 1B, for intermediate node 114, data incoming channel 134 on an upstream side and data outgoing channel 136 on a downstream side together constitute a part of a forward communication channel that is established to pass information along a downstream direction. Similarly, data incoming channel 140 and data outgoing channel 138 together constitute a part of a backward communication channel along an upstream direction. The forward and backward communication channels further pass other successor and predecessor nodes, and extend downstream and upstream across the entire linear communication orbit. Specific information may be communicated successively from node to node along the forward communication channel, the backward communication channel, or both. In some embodiments, the start of the forward communication channel and the end of the backward communication channel further extends to server 132, because head node 118 is also coupled to serve 132 via a bidirectional communication link.

[0069] In various embodiments of the present invention, specific information communicated along the forward or backward communication channel may originate from a server, a head node, a tail node or an intermediate machine (e.g., machine 114 in Figure 1B) of the linear communication orbit. As it is transferred along the forward or backward communication channel, the specific information may be modified at a certain intermediate machine and continue to be communicated to its immediate neighbor machine on the respective communication channel. In some embodiments, an intermediate machine may determine to retain a copy (e.g., locally cache) of the specific information, independently of whether the information needs to be communicated to the neighboring machine. In addition, in some implementations, each machine in the communication orbit may choose to reverse the direction of communication, when the specific information is received at the respective client node. In some implementations, however, such a reversal of direction occurs only at the tail node.

[0070] In accordance with some embodiments of the present invention, specific information that may be communicated in the forward and backward communication channels includes, but is not limited to, system management messages, data requests, specific data (e.g., data requested by the data requests) and status messages. In one example, a system management message is an installation or update message issued by the server, and transferred along the forward communication channel to various client nodes on the linear communication orbit. At a respective client node that receives the installation or update message, the message is parsed into a plurality of data requests each of which may be circulated within the linear communication orbit to request respective specific data. The requested respective specific data are returned to the client node, and distributed to other client nodes along the forward or backward communication channel as well. More details will be presented below concerning specific data caching and distribution processes, such as for installations and updates of applications, databases and the like.

[0071] Figure 2A illustrates an exemplary data caching and distribution system 200 that processes a system management message for information installations or updates in accordance with some embodiments. The data caching and distribution system 200 includes a server 202 and a linear communication orbit 204 that are coupled at a head node 206. In addition to head node 206, the linear communication orbit 204 further includes a tail node 208 and a plurality of intermediate machines 210-218 which are coupled between head node 206 and tail node 208. While intermediate machines 210-218 illustrate the configuration of linear communication orbit 204, in some embodiments, linear communication orbit 204 includes only one intermediate machine, while in other embodiments, it includes dozen, hundreds or even thousands of intermediate machines.

[0072] The system management message is issued by server 202 to request installation or update of the same software application, database, or the like on one or more machines in the linear communication orbit 204. As discussed above, in some implementations, the system management message contains a rule, where the rule is interpreted or evaluated by each computational machine in the linear communication orbit that receives the system management message, and the outcome of that rule interpretation or evaluation by a respective node determines whether node initiates a data transfer to the node, or not. The data transfer, if initiated, typically takes the form of multiple data requests. The

system management message may or may not include an explicit data distribution command. For example, in some embodiments, the system management message includes the identifier of a manifest, or includes the manifest itself. In some embodiments, the manifest is a file or list or other object that includes a list of object shards or other units of data. If the manifest contains information for more than one object, the manifest includes, directly or indirectly, a list of object shards or other units of data corresponding to each object. For example, the manifest can include a list of other manifests, each of which identifies object shards or units of data to be combined to form a respective object. In another example, the set of objects identified or referenced by the manifest includes a first set of files (which may include files having executable programs and/or files containing data), and also one or more installation programs for installing the first set of files.

[0073] If the evaluation of the rule in the system management message by the computation machine (e.g., destination message 214) indicates that the message is applicable to the computation machine, the computational machine issues a request for the manifest, unless the system management message includes the manifest or the computation machine is otherwise already in possession of a copy of the manifest. When the computation machine receives the manifest, or determines it already has it, the computation machine then undertakes to obtain a copy of each object shard or other unit of data referenced by the manifest, excluding those already in its possession. Typically, the computational machine issues multiple requests for specific data items, and then combines the data items listed in or identified by the manifest so as to complete a data distribution task corresponding to the system management message. In effect, in these embodiments, the system management message is a data distribution command, whether or not the system management message contains an explicit data distribution command.

[0074] Stated another way, in accordance with the manifest received by a destination machine, e.g., intermediate machine 214, the destination machine generates a plurality of data requests, each of which is associated with at least one shard of an object to be installed or updated at the destination machine. In one example, each data request is a request of a single respective shard of the object. In some circumstances, one or more shards of the object will have been previously cached in the destination machine and do not need to be requested again. As a result, each destination machine generates a set of data requests to collect only

the shards that are absent from its local cache. Upon acquisition of all the shards corresponding to an object, the destination machine combines the shards to generate the entire object.

[0075] The system management message is typically delivered to multiple computational machines on linear communication orbit 204, which results in concurrent installations or updates of the same file(s), software application(s), database(s) or the like on multiple computational machines on linear communication orbit 204. The system management message is sent along linear communication orbit 204 to computational machines along linear communication orbit 204. As discussed above, the system management message specifies, directly or indirectly (e.g., through the identification of a manifest, or a manifest embedded in the message) one or more objects to be obtained or generated at one or more computational machines in linear communication orbit 204.

[0076] In some embodiments, the system management message originates from server 202 and travels along the forward communication channel that starts from head node 206. At each client node, the respective computational machine determines whether the system management message is applicable to that machine. For example, the computation machines determines whether the message includes a data distribution command that designates the client node, or it evaluates a rule in the message that indicates whether the message is applicable to that computational machine. In accordance with a positive determination, the respective computational machine (e.g., machine 214) retains the corresponding data distribution command, or retains a manifest identified by or embedded in the message. In addition, the system management message continues to be distributed along the forward communication channel to downstream machines for their evaluation. In some embodiments, the intermediate machine 214 generates a command status and reports the command status back to server 202 via the backward communication channel. In some embodiments, when the system management message has been delivered to all the computational machines in the orbit up to tail node 208, it message is deleted by the tail node 208.

[0077] In some embodiments, a system management message originates at a respective node on the linear communication orbit, and is distributed to other nodes (i.e.,

computational machines) along the linear communication orbit starting at the originating node.

[0078] Figure 2B illustrates an exemplary data caching and distribution system 250 that contains a data request path 260 for propagating a data request and requesting specific data in a linear communication orbit in accordance with some embodiments. At a certain machine (e.g., machine 214), a data distribution command or manifest included in (or identified by) a system management message is parsed into a plurality of data requests according to the manifest. Each of the plurality of data requests is passed along the predetermined data request path 260 that tracks the linear communication orbit 204. In this embodiment, data request path 260 originates from computational machine 214 (i.e., a requester machine), extends along the forward communication channel of the linear communication orbit 204, reaches tail node 208, reverses to follow the backward communication channel, and terminates at head node 206.

[0079] In some embodiments, each data request is passed along the data request path 260, until either (A) a cached copy of the specific data (e.g., a shard of a data set) is identified within another computational machine on the data request path 260, or (B) the data request reaches head node 206 because no machine located on the data request path 260 is found to store a cached copy of the specific data in its local cache. In the latter case, head node 206 further passes the data request to server 202 to request a copy of the specific data from the server, and the data request path 260 extends from head node 206 to server 202.

[0080] When the requester machine is head node 206, the data request path 260 follows the forward (downstream) and backward (upstream) communication channels, as needed, prior to potentially extending to server 202, if needed. When the requester machine is tail node 208, the data request path tracks the backward (upstream) communication channel and reaches head node 206 if none of the intermediate nodes store a cached copy of the specific data in its local cache.

[0081] Once the computational machines along the linear communication orbit 204 have received a system management message that includes a manifest or one or more data distribution commands, server 202 has no further involvement in the distribution and installation of the object(s) specified by the manifest or data distribution commands, other than providing shards to head node 206 when such shards are not found in the local cache of

any of the computational machines along linear communication orbit 204. From the perspective of each intermediate machine, the intermediate machine communicates only with its immediate neighbors when responding to a data distribution command in order to obtain the shards corresponding to (or specified by) the data distribution command.

[0082] In some embodiments, each intermediate machine on data request path 260 maintains a respective tracking record for each distinct shard that it has requested, or that is requested by a data request received by the intermediate machine. When a data request is received for a shard for which the intermediate node has a tracking record, the local cache of the intermediate machine is not searched for a second time, if the tracking record indicates that the local cache has already been searched for the shard and does not have a copy of the shard. The corresponding data request is directly passed to the next immediate machine on data request path 260 without repeating the search for the shard. Such a tracking record is particularly useful when an intermediate machine has recently searched the local cache for the requested data in response to one data request and then receives another data request that requests the same data again. In some implementations, the tracking record remains valid for a limited duration of time, and is reset or cleared upon receiving a response to the corresponding request.

[0083] While Figures 2A-2B and other figures presented here show messages and data being communication only along forward and reverse paths of a linear communication orbit 204, with each node sending and receiving messages only from its immediate neighbors, the methods described herein are also applicable to methods where messages are conveyed with the linear communication orbit along other paths. For example, when a node in the linear communication orbit becomes inoperative or unresponsive, messages may be conveyed within the linear communication orbit through an alternative path, so was to skip over or otherwise work around the inoperative or unresponsive node. In such circumstances, the immediate neighbors of a respective node are the nodes before and after it, along a modified data request path within the linear communication orbit.

[0084] Figures 3A-3C illustrate three respective linear communication orbits 310 that are associated with three scenarios of transferring a data request and distributing specific data based on a location of the specific data in accordance with some embodiments. Each of data requests 302 in Figures 3A-3C originates from a requester machine 304, and travels along at

least a part of data request path 260 as shown in Figure 2B. In response to data request 302, the specific data are identified in a data machine 312, and returned to requester machine 304 along a data return path 320 that varies according to the location of data machine 312. In some embodiments, data return path 320 extends from the location of the specific data to one of two end nodes, including tail node 306 and head node 308.

[0085] In some embodiments as shown in Figure 3A, data request 302A is transferred along a part of the forward communication channel on linear communication orbit 310. A cached copy of the specific data is identified in data machine 312A located downstream of requester machine (M1) 304A, and therefore, the data request ceases to be passed to machines on the rest of data request path 260. In this circumstance, the specific data are returned along a data return path 320A that extends between data machine (M2) 312A and head node 308 and overlaps a part of the backward communication channel of linear communication orbit 310.

[0086] In some embodiments as shown in Figure 3B, data request 302B is transferred along both a part of the forward communication channel and a part of the backward communication channel on linear communication orbit 310. A cached copy of the specific data is identified in a data machine (M2) 312B located within linear communication orbit 310 and upstream of requester machine (M1) 304B. As a result, the data request is not passed further upstream beyond data machine (M2) 312B, and the specific data is returned along a data return path 320B that extends between data machine 312B and tail node 306 and overlaps a part of the forward communication channel of linear communication orbit 310.

[0087] In some embodiments as shown in Figure 3C, data request 302C is transferred along entire data request path 260 until it reaches head node 308. In this example, no cached copy of the specific data is identified in any computational machine within linear communication orbit 310. Therefore, head node 308 passes data request 302C to server 314 which provides the requested specific data to head node 308. In this circumstance, once it receives the specific data from server 314, head node 308 is regarded as a data machine 312C within linear communication orbit 310 that has a copy of the requested specific data. The specific data is then returned along a data return path 320C that extends between head machine 308 and tail node 306 and overlaps the entire forward communication channel of linear communication orbit 310.

[0088] In some embodiments, data machine 312 includes a machine (within linear communication orbit 310) that is located closest, along the data request path, to requester machine 304 and has a cached copy of the specific data in its local cache.

[0089] In some embodiments as shown in Figures 3A-3C, requester machine 304 and data machine 312 are separated by at least one intervening machine (M3) 316 on linear communication orbit 310. Independently of the location of data machine (M2) 312, the specific data are successively passed along data return path 320 by the intervening machines coupled between data machine (M2) 312 and requester machine (M1) 304, until they are received by requester machine 304. In addition, each intervening machine determines whether a copy of the specific data is to be cached in its local cache as well. Upon receiving the specific data from its immediate neighbor, requester machine 304 stores a copy of the specific data, and marks the data request as completed.

[0090] In some embodiments, after they have been delivered to requester machine 304, the specific data are further passed to additional machines 318 located on data return path 320, until it is received by the corresponding end node, head node 308 or tail node 306, at the end of data return path 320. In some implementations, each additional machine 318 that receives the specific data also determines whether a copy has already been retained locally and, if not, whether such a copy of the specific data is to be locally cached.

[0091] In some embodiments, additional machine 318 determines whether a copy has already been retained locally by checking to see if it has a tracking record for a shard, which indicates whether the additional machine's local cache has already been searched for corresponding specific data. If the tracking record is found, the specific data are not present in the local cache. If a tracking record for the shard is not found, additional machine 318 searches its local cache to determine whether a cached copy has already been retained in its local cache.

[0092] In some embodiments, a copy of the specific data is created in a local cache of an intervening or additional machine when certain predefined local caching criteria are satisfied. In a broad perspective, the predefined local caching criteria include that no copy of the specific data is already stored in the respective cache as discussed above. In some embodiments, the caching criteria require that certain data not be cached in certain machines due to various reasons (e.g., data security). In some embodiments, the caching criteria set a

limit to the number of copies that may be cached in linear communication orbit 310, or prohibit the specific data from being cached in two immediately adjacent machines.

[0093] When the predefined local caching criteria are met, the copy of the specific data is locally cached according to a data caching method. The data caching method may include a statistically random or pseudorandom method of determining whether to cache respective data received from another machine. In some embodiments, a probability of caching specific data received at any intervening or additional machine is determined in accordance with a parameter, such as the number of machines included in linear communication orbit 310. In one example, the probability of caching has a default value of N/T , where N is an integer corresponding to a target number of copies to be cached within a linear communication orbit, and T is an integer corresponding to the total number of machines in the linear communication orbit. In another example, the probability of caching has a default value of $1/L$, where L is an integer (e.g., an integer in the range of 6 to 12). Optionally, the default value for the probability of caching is used when the linear communication orbit 310 has at least a predefined number of machines. However, in some implementations, the probability of caching is reduced to a predefined value, such as $1/4$, when the number of machines in linear communication orbit 310 is less than another integer M , e.g., 8.

[0094] In some embodiments, the probability of caching specific data (e.g., a shard of a file or data set) at any specific machine is determined at the specific machine in accordance with at least one of: a size of the specific data, a total size of a corresponding object or dataset that includes the specific data, a size of the local cache of the specific machine, a size of the unused space in local cache of the specific machine, and a size of all local caches on linear communication orbit 310. In one example, at a machine having a local cache size of 100 Mbytes, the probability to caching a shard of a data set having a total size of 1.0 GBytes is ten percent (10%). In another example, at a machine having a local cache size of X , the probability to caching a shard of a data set having a total size of Y is X/Y (e.g., 10%) so long as the cache has sufficient room to store the shard without evicting from the cache any data that meets predefined non-eviction criteria (e.g., the machine may have predefined non-eviction criteria that prevents eviction of “recently received or used” shards from the same data set as the shard being considered for eviction). In some implementations, the non-

eviction criteria are: specific data (sometimes herein called a shard for ease of explanation) is not to be evicted from the cache if either the specific data was first stored in the cache in the last N seconds, where N is a predefined value between 1 and 60, or the specific data was most recently sent to another machine, in response to a data request, in the last M seconds, where M is a predefined value between 1 and 60, where M is not necessarily the same value as N.

[0095] In some other embodiments, the cached copies are distributed substantially evenly across linear communication orbit 310. For example, a predefined machine distance is maintained between machines (in the linear communication orbit) selected to store copies of the specific data.

[0096] During the course of caching and distributing the specific data, a caching status message may be generated to track copies the specific data in the communication orbit and used to report a caching status back to server 314. In some embodiments, when the specific data are returned to requester machine 304 or cached in a certain intervening or additional machine, the respective machine generates a distinct caching status message and reports it separately to server 314. In some other embodiments, a caching status message traverses the machines of the linear communication orbit, or a subset of the machines of the linear communication orbit, collecting information identifying the machines at which the specific data are locally cached. At the head or tail node, the resulting caching status message is sent to server 314.

[0097] In some embodiments, machines in linear communication orbit 310 are included in (and thus interconnected by) a local area network, and communicate with server 314 via head node 308, which is coupled to server 314 through a wide area network. As shown in Figures 3A-3C, the machines in the linear communication orbit 310, coupled via the local area network, are searched for specific data. This is called a local search. Only when the local search fails, the corresponding data request is sent to server 314, accessed via the wide area network. Moreover, the specific data are cached and distributed by machines on linear communication orbit 310 during the course of delivering the specific data to a requester machine 304. As a result, extra copies of the specific data are locally cached at additional machines to facilitate data distribution to other machines requesting the same specific data, using only the local area network. When multiple machines on the same linear communication orbit 310 request the same specific data within a short period of time (e.g., a

period of less than an hour, or less than 10 minutes), the above data caching and distribution method, on average, substantially reduces data transmitted via the wide area network from server 314 compared with methods in which each machine on the local area network requests data from and receives the requested data from a server via a wide area network.

[0098] Figure 4 illustrates an exemplary process 400 of providing the data requested by two respective data requests, both generated by a computational machine 402 (M1) on a linear communication orbit 410 in accordance with some embodiments. In this example, according to a manifest received from another computational machine in the same linear communication orbit 410 or from a server, computational machine 402 (M1) needs to obtain a plurality of data, including first data and second data, that together constitute an object to be installed or updated on computational machine 402. Therefore, computational machine 402 generates a plurality of data requests, including a first data request and a second data request, for requesting some of the plurality of data from other machines on linear communication orbit 410. Each data request is sent by a computational machine, e.g., requester machine 402, and is transferred along a respective data request path. In response to each data request, the corresponding data are identified in a computational machine on linear communication orbit 410, e.g., machine 404 (M2) or 406 (M4), and returned to requester machine (M1) 402 along a respective data return path. More details on data requests, data caching and data distribution associated with an individual data request are explained above with reference to Figures 3A-3C.

[0099] In some embodiments, although the two data requests originate from the same requester machine (M1) 402, their corresponding data are identified in different computational machines on linear communication orbit 410. For example, the first data requested by the first data request are identified in computational machine 406 that is located downstream of machine (M2) 404, which stores a copy of the second data requested by the second data request. In this instance, the resulting data return paths for the first and second data originate differently, but both extend along the backward communication channel. In some embodiments, the data return paths terminate at a head node 408 of linear communication orbit 410, thereby giving machines in the linear orbit along the data return paths the opportunity to store a cached copy of the data requested by the first machine (M1). Although they may not entirely overlap, the data return paths for requested data of machine

402 pass requester machine 402. In some alternative embodiments, or in some circumstances, the data return paths terminate at the requesting node 402.

[00100] At the end of the data caching and distribution process, requester machine 402 collects the plurality of data that include shards in the object to be installed or updated. Each of the data is either identified in the local cache of machine 402 or returned to machine 402 via the corresponding data return path. Requester machine 402 then combines the plurality of data to generate or reconstitute the object according to the manifest received together with the original system management message or data distribution command.

[00101] In many embodiments, multiple machines, including machine 402, on linear communication orbit 410, based on the same manifest, concurrently (i.e., during overlapping time periods) generate requests for the same shards of the same object that needs to be installed or updated on these computational machines. As a result, machines in the linear communication orbit 410 could be flooded with a large number of data requests in a short period of time. Therefore, in some embodiments, at least some of these machines delay generating their respective data requests for shards to avoid generating data requests for the same specific data within a short period of time. This allows local caches of machines on linear communication orbit 410 to be gradually populated with the requested shards.

[00102] In some embodiments, at least some of the data requests generated by one requester machine are delayed before they are sent to request shards included in the object from machines on linear communication orbit 410. In some embodiments, this process, sometimes herein called data throttling, is invoked only when (i.e., in accordance with a determination that) a respective machine has generated and/or passed along to neighboring nodes at least a threshold number (e.g., a number between 4 and 20) of unanswered data requests. In some embodiments, different requester machines on linear communication orbit 410 use different length delays between successive data requests generated, or passed along. As a result, data requests generated by different machines are transmitted and processed in a non-concurrent manner in linear communication orbit 410. In some embodiments, the lengths of the delays between successive requests sent by a respective machine vary in accordance with a predefined methodology. In some embodiments, the lengths of the delays between successive requests transmitted by a respective machine fall within a predefined range, such as 0.25 seconds to 5 seconds.

[00103] In some embodiments, one or more respective machines on linear communication orbit 410 limit the number of outstanding (i.e., unanswered) data requests (whether generated by the respective machine or passed along) to a predefined number (e.g., a number between 4 and 20). In these embodiments, as requested data arrives at the requested machine, if additional data requests are pending, those data requests are generated or passed along to neighboring machines, subject to the limit on outstanding data requests.

[00104] Figure 5 illustrates an exemplary process 500 of consolidating data requests issued by two distinct computational machines to request the same specific data on a linear communication orbit 510 in accordance with some embodiments. When a server sends a system management message or command that results in an identical object being installed or updated on more than one computational machine on linear communication orbit 510, different computational machines (e.g., machines (M1) 502 and (M5) 504) issue respective data requests to request the same specific data that constitute a shard of the object. In some implementations, the respective data requests are generated in an asynchronous manner, for example in accordance with times introduced according to the data throttling method (sometimes called a spreading method) described above. Consolidation of such data requests helps avoid repetitive data request processing and redundant data caching.

[00105] In one example, requester machines (M1) 502 and (M5) 504 that request the same specific data receive a copy of the specific data from the same data machine 506, and share at least a part of their respective data request path. In some embodiments, data machine (M2) 506 is located downstream of both machines (M1) 502 and (M5) 504, while in other embodiments, data machine 506 is located upstream of both machines 502 and 504. However, in some embodiments, data machine (M2) 506 is located between machines (M1) 502 and (M5) 504, in which case machines 502 and 504 have non-overlapping data request paths and each directly obtains the copy of the specific data from data machine 506.

[00106] In some implementations, a computational machine consolidates the data requests that separately originate from machines (M1) 502 and (M5) 504, when it determines that predefined criteria in a request consolidation method are satisfied for the data requests. In one example, the computational machine is one of requester machine 502, requester machine 504 or an intermediate machine that is located on an overlapping part of the corresponding data request paths of the data requests. Upon a determination that the criteria

in the request consolidation method are met, the computational machine forgoes passing one of the data requests along at least a part of its respective data request path.

[00107] In accordance with some exemplary criteria in the request consolidation method, the computational machine determines whether another data request has been previously generated or processed to request the same specific data during a predetermined period of time. The period of time expires when the corresponding specific data are returned, or an expiration time (e.g., a predetermined amount of time after the request is generated or transferred to a next machine along the data request path) is reached.

[00108] As shown in Figure 5, two computational machines (M1) 502 and (M5) 504 respectively generate a first data request and a third data request to request the same specific data, e.g., a shard of an object. For either machine 502 or machine 504, a respective delay is optionally applied between a first time when a data distribution command or system management message having (or identifying) a manifest is received from the server, and a second time when the corresponding data request is generated. In some circumstances, the first data request is generated at machine 502 after the third data request has been generated by machine 504. Thus, when the first data request is generated at machine 502, the third data request has already been passed by machine 502 to a next machine on the data request path, and machine 502 retains a corresponding tracking record. Thus, when the first data request is generated, machine 502 detects the retained tracking record for the third data request, and based on that information, forgoes sending the first data request to a next machine on the data request path. Machine 502 waits for the corresponding specific data to be returned in response to the third data request, and uses the same specific data to satisfy the first data request.

[00109] In certain circumstances, the third data request is generated at machine 504 after the first data request has been generated by machine 502. Machine 502 retains a tracking record corresponding to the first data request when it sends the first data request to a next machine along the data request path. Based on the tracking record for the first data request, the third data request is not passed by machine 502 to a next machine along the data request path.

[00110] More generally, each machine that issues or passes a data request retains a tracking record of the request until a response with the request data is received, at which

point the tracking record is either removed or marked as no longer pending. If a subsequent request is received after the earlier request has received a response, then the processing of the later data request is not impacted by the earlier request, except that the specific data required to respond to the later request is likely to be locally cached by at least one machine in the linear communication orbit 510.

[00111] Figure 6A is a block diagram of an exemplary computational machine 600 in the exemplary linear communication orbits shown in Figures 1A-5 in accordance with some embodiments. In some implementations, machine 600 includes one or more processors 602a, memory 604a for storing programs and instructions for execution by one or more processors 602a, one or more communications interfaces such as input/output interface 606a and network interface 608a, and one or more communications buses 610a for interconnecting these components.

[00112] In some embodiments, input/output interface 606a includes a display and input devices such as a keyboard, a mouse or a track-pad. In some embodiments, communication buses 610a include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. In some embodiments, memory 604a includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and optionally includes non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. In some embodiments, memory 604a includes one or more storage devices remotely located from the one or more processors 602a. In some embodiments, memory 604a, or alternatively the non-volatile memory device(s) within memory 604a, includes a non-transitory computer readable storage medium.

[00113] In some embodiments, memory 604a or alternatively the non-transitory computer readable storage medium of memory 604a stores the following programs, modules and data structures, instructions, or a subset thereof:

- Operating System 612a that includes procedures for handling various basic system services and for performing hardware dependent tasks.
- I/O module 614a that includes procedures for handling various basic input and output functions through one or more input and output devices.

- Communication module 616a that is used for connecting machine 600 to other machines (e.g., other machines 600 in network 100) or servers (e.g., server 108) via one or more network communication interfaces 608a (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on.
- Orbit formation module 618a that includes instructions implementing a predetermined set of rules for creating, maintaining, and repairing the linear communication orbit for network and system management.
- Message and command module 620a that includes instructions for handling receipt, processing, propagation, collecting and reporting of system, security and network management messages and commands.
- Neighboring node information 622 that includes information identifying neighboring nodes of machine 600.
- Messages, reports and/or other data 624a that is stored, temporarily or otherwise, upon receipt from a predecessor node, successor node or server, and/or that is locally generated, revised or supplemented by machine 600 prior to transmission to a predecessor node, successor node or server.
- Data distribution module 626a that includes instructions for handling receipt, processing, propagation, collecting and reporting in response to system management messages for distributing objects to be installed or updated on machine 600. In some embodiments, the data distribution module 626a is included in the message and command module 620a. In some embodiments, the data distribution module 626 extracts a manifest (or a data distribution command or manifest identifier) from the system management message received from a server, generates a plurality of data requests, identifies or receives respective specific data in response to each data request, and generates or reconstitutes the object from the identified or received specific data according to the manifest.
- Data request module 628 that includes instructions for processing data requests received from a predecessor node, successor node or server. The data request module 628 determines whether a copy of specific data is available in machine 600. In

accordance with a determination result, the data request module 628 returns the copy to a previous machine or further sends a data request to a subsequent machine on the data request path. In some circumstances, the data request module 628 forgoes sending a data request to successor node or predecessor node, thereby consolidating data requests issued by different machines to request the same specific data.

- Data caching module 630 that includes instructions for receiving, caching, and propagating specific data that pass machine 600 and for reporting related caching activities at machine 600. To be specific, the data caching module 630 determines whether specific data shall be cached in a local cache 632 according to a data caching method and caching the specific data in accordance with a positive determination of data caching.
- Local cache 632.

[00114] Figure 6B is a block diagram of an exemplary server 650 in the exemplary linear communication orbits shown in Figure 1A-5 in accordance with some embodiments. In some implementations, server 108 includes one or more processors 602b, memory 604b for storing programs and instructions for execution by the one or more processors 602b, one or more communications interfaces such as input/output interface 606b and network interface 608b, and one or more communications buses 610b for interconnecting these components.

[00115] In some embodiments, input/output interface 606b includes a display and input devices such as a keyboard, a mouse or a track-pad. In some embodiments, communication buses 610b include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. In some embodiments, memory 604b includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and optionally includes non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. In some embodiments, memory 604b includes one or more storage devices remotely located from the one or more processors 602b. In some embodiments, memory 604b, or alternatively the non-volatile memory device(s) within memory 604b, includes a non-transitory computer readable storage medium.

[00116] In some embodiments, memory 604b or alternatively the non-transitory computer readable storage medium of memory 604b stores the following programs, modules and data structures, instructions, or a subset thereof:

- Operating System 612b that includes procedures for handling various basic system services and for performing hardware dependent tasks.
- I/O module 614b that includes procedures for handling various basic input and output functions through one or more input and output devices.
- Communication module 616b that is used for connecting server 650 to machines 600 coupled to network 100 via one or more network communication interfaces 608b (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on.
- Orbit formation module 618b that includes instructions to determine which machines 600, of all machines 600 currently known to be coupled to network 100, are coupled to the same local area network, and to communicate information to those machines to enable them to self-organize into a linear communication orbit. In some embodiments, orbit formation module also stores a list of singletons, and head nodes and/or tail nodes of the linear communication orbits in the network.
- Message and command module 620b that includes instructions for providing and collecting system, security and network management messages. In some embodiments, message and command module 620b provides a user interface for a network or system administrator to directly perform various system and network functions, such as issuing status inquiries, providing management instructions, and deploying system configurations.
- Network node information 626 that includes information identifying all nodes known to be coupled to network 100. In some embodiments, server 650 maintains a record of nodes currently known to be coupled to the managed network and their respective unique identifiers (e.g., IP addresses and optionally other unique identifiers). Optionally, server 650 also maintains the ordinal positions of these machines in a sorted sequence according to their respective unique identifiers.

- Head, tail and singleton node information 629, identifying head nodes, tail nodes and singleton nodes with established communication channels to and/or from server 650. In some embodiments, server 650 maintains a list of head nodes, tail nodes, and singleton nodes that have opened forward and backward connections with server 650.
- Messages, reports and/or other data 624b that is temporarily stored upon receipt from a head node, tail node, or other reporting node.
- Data distribution module 626b that includes instructions for providing a particular system management message and collecting status messages for distribution and caching of an object. In some embodiments, the data distribution module 626b is included in the message and command module 620b. Specifically, the data distribution module 626b generates a system management message that includes one or more data distribution commands and/or manifests for distributing an object to at least one computational machine on a linear communication orbit. Optionally, the linear communication orbit returns at least one caching status message to server 650, and the data distribution module 626b tracks locations on the linear communication orbit of locally cached copies of each shard included in the object. In some embodiments, the message and command module 620b also provides a user interface for a network or system administrator to dispatch software installation or update commands, or database installation or update commands, to a selected group of machines.

[00117] Figures 6A and 6B are merely illustrative of the structures of machines 600 and server 650. A person skilled in the art would recognize that particular embodiments of machines 600 and server 650 may include more or fewer components than those shown. One or more modules may be divided into sub-modules, and/or one or more functions may be provided by different modules than those shown. In some embodiments, an individual one of machines 600 implements or performs one or more methods described herein as being performed by a respective computational machine, including the methods described with respect to Figures 11A-11D. In some embodiments, a plurality of machines 600 in a linear communication orbit, such as those shown in Figures 1A through 5, together implement or perform one or more methods described herein as being performed by a plurality of computational machines, including the methods described with respect to Figures 12A-12G.

[00118] Figure 7 illustrates an exemplary computational machine 700 that responds to an incoming data distribution command in accordance with some embodiments. In some embodiments, machine 700 receives a system management message from another machine, herein called a first machine, and evaluates the system management message, to determine if the system management message is applicable to machine 700. In these embodiments, the system management message includes the aforementioned data distribution command. If machine 700 determines that the received system management message is applicable to machine 700, machine 700 then executes the aforementioned data distribution command.

[00119] For example, when machine 700 is an intermediate machine on a linear communication orbit, the first machine is typically an immediate neighbor machine that is located upstream of machine 700. When machine 700 is a head node on the linear communication orbit, the first machine is a server that is coupled to the linear communication orbit via the head node. In accordance with a manifest in, or specified by, the system management message, the data distribution command is parsed into a plurality of data requests to request shards of an object that needs to be installed or updated on machine 700. Each shard is returned to machine 700 either from the first machine or from a second machine on the linear communication orbit, such as an immediate neighbor machine located downstream of machine 700 on the linear communication orbit.

[00120] In some embodiments, other than evaluating the received system management message to determine if the message is applicable to machine 700, and generation of the data requests, machine 700 does not proactively control search, distribution and caching of copies of the shards included in the object; rather, machine 700 passively waits for the requested data to be returned by neighboring machines in the linear communication orbit (e.g., either machine of its two immediate neighbor machines). In some embodiments, when the requested data are not returned within a predetermined wait time, machine 700 sends a status message to a server via the first immediate neighbor machine, and the server provides a copy of the requested data to machine 700 via the first immediate neighbor machine.

[00121] Figure 8A illustrates an exemplary intermediate machine 800 that processes a data request and conditionally stores specific data requested by the data request in accordance with some embodiments. Intermediate machine 800 receives the data request from a first machine, and in various embodiments, the first machine may be located upstream or

downstream of machine 800 on a linear communication orbit. The data request is generated by the first machine or another machine that is located further away along the stream. In response to this data request, intermediate machine 800 determines whether the requested specific data are stored in its local cache. In accordance with a determination by intermediate machine 800 that the specific data are stored in its local cache, the specific data are returned by intermediate machine 800 to the first machine, and the data request is not passed to any other machines by intermediate machine 800.

[00122] In contrast, when intermediate machine 800 determines that the specific data are not stored in its local cache, the data request is passed by intermediate machine 800 to a second machine that is distinct from the first machine. The second machine or each machine on a corresponding data request path successively repeats the aforementioned determination process, described here for intermediate machine 800, until a copy of the specific data is identified and successively returned to intermediate machine 800. Then, intermediate machine 800 returns the specific data to the first machine, and moreover, determines whether to store the specific data in the local cache of intermediate machine 800 according to a data caching method as explained above.

[00123] In some embodiments, the requested specific data are not identified in machines included in the corresponding linear communication orbit, and have to be requested from a server via a head node. Figure 8B illustrates an exemplary computational machine 850 that acts as a head node of a linear communication orbit and conditionally stores specific data during the course of processing a corresponding data request in accordance with some embodiments. Head node 850 receives a data request from a first machine that is an immediate neighbor machine included in the linear communication orbit. In some circumstances, head node 850 identifies the requested specific data in its local cache and returns the requested specific data to the first machine. However, in many circumstances, head node 850 does not have the requested specific data, and therefore, passes the data request to a server 860 that is coupled to the linear communication orbit. When server 860 returns a copy of the specific data, head node 850 returns the copy to the first machine, and determines whether to store the specific data in the local cache of head node 850 according to a data caching method as explained above.

[00124] In some embodiments, the local cache of any particular machine on a linear communication orbit is a part of the entire memory system of the particular machine. In some implementations, contents in the local cache are evicted according to a respective predetermined eviction policy. In some embodiments, the predetermined eviction policy specifies a predetermined and fixed eviction time, and requires the specific data be deleted from the local cache after the predetermined eviction time passes. However, in some other embodiments, caching activities are monitored for each item (e.g., shard, manifest, file or the like) stored in the local cache. When space is needed in the local cache to store new content, one or more items in the local cache is deleted in accordance with the monitored caching activities (e.g., one or more cached items deemed to be the most stale are deleted from the local cache).

[00125] Under many circumstances, each computational machine on the linear communication orbit independently manages the handling of data requests and the handling of specific data received at the computational machine. In particular, upon receiving a data request, each computational machine proactively processes the data request according to a set of predetermined routines and the internal state of the computational machine, without requiring instruction or state information from any other computation machine. Similarly, upon receiving specific data, each computational machine proactively processes the specific data according to a set of predetermined routines and the internal state of the computational machine, without requiring instruction or state information from any other computation machine. As a result, in many situations, the data request and distribution process is self-organized, self-managed and completed locally on the linear communication orbit.

[00126] Figure 9 illustrates an exemplary computational machine 900 that consolidates two data requests that request the same specific data in accordance with some embodiments. The two data requests include a first data request and a second data request in which the first data request is processed by machine 900 at a time earlier than the second data request. The first data request is either generated at machine 900 or passed from a first machine to a second machine via machine 900. In some embodiments, the first machine and the second machine constitute two immediate neighbor machines of machine 900. In response to the first data request, specific data are returned from the second machine to machine 900.

[00127] The second data request is generated at machine 900 or received from either one of the first and second machines. Under some circumstances, the first and second data requests are received from different machines of the first and second machines. In addition, the second data request is generated or received at machine 900 before the requested specific data are returned to machine 900 in response to the first data request.

[00128] The first and second data requests are consolidated at machine 900 when predefined criteria in a request consolidation method are satisfied. As a specific example, if no response to the first data request has yet been received by machine 900, machine 900 forgoes passing the second data request to a subsequent machine when that would require machine 900 to pass the second data request to the same subsequent machine as the machine to which it passed the first data request. When the requested specific data are returned to machine 900 in response to the first data request, machine 900 responds to both the first data request and the second data request. If either request was generated by machine 900, that request is “responded to” by retaining the specific data. For example, the specific data may be passed, within machine 900, to an application that will combine the specific data with other data so as to generate an object. As a result, the first and second data requests are consolidated at machine 900. More details on the data request consolidation process are explained from the perspective of requester machines in the above description of Figure 5.

[00129] Figure 10A illustrates a conceptual block diagram of a linear communication orbit 1000 that distributes and caches specific data in accordance with some embodiments. Linear communication orbit 1000 includes a requester machine 1002, an intermediate machine 1004 and a data machine 1006. Requester machine 1002 generates a data request that requests specific data, and the specific data are subsequently identified in data machine 1006. As the specific data are returned to requester machine 1002 from data machine 1006, they pass intermediate machine 1004 and are conditionally stored within a local cache of intermediate machine 1004 based on a data caching method.

[00130] Linear communication orbit 1000 terminates at two end nodes, including a tail node and a head node that is coupled to a server. In some embodiments, requester machine 1002, data machine 1006 or both of them are the end node(s) of linear communication orbit 1000.

[00131] In some embodiments, a data request is passed along a forward communication channel from requester machine 1002 to data machine 1006, in a downstream direction on linear communication orbit 1000, and the specific data are returned upstream from data machine 1006 to requester machine 1002. However, in some circumstances (e.g., when none of the machines downstream of machine 1004 have a locally cached copy of the data requested by a data request, the data request is passed downstream from requester machine 1002 to a tail node, and returned upstream to data machine 1006. In response to the data request, the specific data are returned downstream from data machine 1006 to requester machine 1002.

[00132] When intermediate machine 1004 receives the specific data requested by a data request, it independently determines whether to cache the specific data according the data caching method as explained above, without regard to the locations of requester machine 1002 and data machine 1006.

[00133] Figure 10B illustrates a conceptual block diagram of another linear communication orbit 1010 that distributes and caches specific data in accordance with some embodiments. In this example, requester machine 1002 is located downstream of intermediate machine 1004 and head node 1008, and in some embodiments, requester machine 1002 is the tail node of linear communication orbit 1010. In some circumstances, none of the machines included in linear communication orbit 1010 has a copy of the specific data, and therefore, when head node 1008 receives the data request, it requests the specific data from a server. When head node 1008 receives the specific data from the server, it functions similarly as data machine 1006 in Figure 10A by returning the specific data to the next machine downstream of head node 108.

[00134] Therefore, as shown in Figures 10A-10B, when head node 1008 receives a data request generated by requester machine 1002, it functions as data machine 1006 by either providing a copy of the requested data that is already stored in its local cache or requesting a copy of the requested data from a server and then providing to requester machine 1002 the copy of the requested data received from the server (by forwarding the data to the next machine in the downstream direction).

[00135] Figures 11A-11D include a flow diagram representing a method 1100 for caching and distributing specific data in accordance with some embodiments. Method 1100

is, optionally, governed by instructions that are stored in a non-transitory computer readable storage medium and that are executed by one or more processors of a computational machine (e.g., machine 600). Each of the operations shown in Figures 11A-11D may correspond to instructions stored in a computer memory or non-transitory computer readable storage medium (e.g., memory 604a of machine 600 in Figure 6A). The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The instructions stored on the computer readable storage medium may include one or more of: source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors. Some operations in method 1100 may be combined and/or the order of some operations may be changed.

[00136] Method 1100 is performed by a computational machine that has a local cache. The computational machine receives (1102) from a first machine a data request that is used to request specific data. In some embodiments, the specific data comprises (1104) one of a plurality of shards of an object, such as a software application, software application update, software patch, database, or database update. In some implementations, the computational machine and the first machine are interconnected (1106) by a single local area network. In some implementations, the computational machine and the first machine are two machines of a set of machines in a linear communication orbit, all of which are interconnected by a single local area network.

[00137] After receiving the data request, the computational machine determines (1108) whether the computational machine stores the specific data in the local cache of the computational machine. When the computational machine determines that the specific data are not stored in its local cache, it responds (1110) to the data request by passing (1112) the data request to a second machine. Additionally, in response to the data request, when the second machine returns the specific data to the computational machine, the computational machine returns (1114) the specific data to the first machine, and also determines (1114) whether to store the specific data in its local cache according to a data caching method.

[00138] In some embodiments, the data caching method includes (1116) a statistically random method of determining whether to cache respective data received from another machine. In some embodiments, the data caching method is determined (1118) based on at

least one of the size of the specific data, the size of the local cache, the size of available caches in a linear communication orbit that includes the computational machine, and the size of a respective data set (e.g., an object, or a set of objects) that includes the specific data. As a specific example, a probability of caching the specific data at the computational machine is determined in accordance with a ratio between the size of the local cache and the size of an object, of which the requested specific data is a shard. In another example, a probability of caching the specific data at the computational machine is determined in accordance with a ratio between the size of the local cache and the size of the requested specific data. In either of these examples, larger local caches have higher probability of storing a particular specific data (e.g., shard) than smaller local caches. Additional details and examples of the data caching method are explained above with reference to Figures 3A-3C.

[00139] In some circumstances, when the computational machine determines (1120) that the specific data are stored in its local cache, it responds to the data request by returning the specific data to the first machine. Stated another way, it responds by a sending a copy of the specific data, obtained from its local cache, to the first machine.

[00140] Under some circumstances, the data request comprises (1122) a first data request, and method 1100 includes additional operations to consolidate the first data request with a second data request. To be specific, the computational machine receives (1124) the second data request, after receiving the first data request, to request the same specific data as the first data request. For example, the second data request is initiated (1126) by a third machine distinct from the first machine.

[00141] After receiving the second data request, the computational machine determines whether predefined criteria in a request consolidation method are satisfied. In accordance with a determination that the criteria are satisfied, the computational machine forgoes (1128) passing the second data request to the second machine. In other words, the second data request is consolidated with the first data request that requests the same specific data by the computational machine.

[00142] From another perspective, not reflected in Figures 11A-11D, request consolidation is performed at a computational machine having a local cache when, after generating a first data request that is used to request specific data, and passing the data request to a second machine, the computation machine receives, from a third machine, a

second data request to request the same specific data. In this method, in accordance with a determination that predefined criteria are satisfied (e.g., that the first data request is still outstanding, waiting for a response), the computational machine forgoes passing the second data request to the second machine. Furthermore, when the second machine returns the specific data to the computational machine, the computational machine returns the specific data to the third machine. Thus, the second data request is consolidated at the computational machine which earlier generated a data request for the same data as the later received second data request.

[00143] In some embodiments, the computational machine is (1130) is a head node of a linear communication orbit that is organized from a plurality of machines, including the computational machine and the first machine, based on a set of rules at a specific time. In this circumstance, the second machine is a server coupled to the head node.

[00144] In some embodiments, the computational machine is (1132) an intermediate machine between neighboring machines within a linear communication orbit. The linear communication orbit is organized from a plurality of machines based on a set of rules at a specific time and subject to changes in configuration and membership when one or more machines join or exit the linear communication orbit. In one specific example, the computational machine is (1134) a downstream of the first machine and upstream of the second machine in the linear communication orbit. In another specific example, the computational machine is (1136) a downstream of the second machine and upstream of the first machine in the linear communication orbit. For corresponding examples, see Figures 3A and 3B and the description above of these figures.

[00145] It should be understood that the particular order in which the operations in Figures 11A-11D have been described are merely exemplary and are not intended to indicate that the described order is the only order in which the operations could be performed. One of ordinary skill in the art would recognize various ways to cache and distribute specific data as described herein. Additionally, it should be noted that details of other processes described herein with respect to method 1200 (e.g., Figures 12A-12G) are also applicable in an analogous manner to method 1100 described above with respect to Figures 11A-11D. For brevity, these details are not repeated here.

[00146] Figures 12A-12G include a flow diagram representing another method for caching and distributing specific data in accordance with some embodiments. Method 1200 is, optionally, governed by instructions that are stored in a non-transitory computer readable storage medium and that are executed by one or more processors of a plurality of computational machines (e.g., machine 600). Each of the operations shown in Figure 12 may correspond to instructions stored in the computer memory or computer readable storage medium (e.g., memory 604a of machine 600 in Figure 6A) of each of the plurality of computational machines. The computer readable storage medium may include a magnetic or optical disk storage device, solid state storage devices such as Flash memory, or other non-volatile memory device or devices. The computer readable instructions stored on the computer readable storage medium may include one or more of: source code, assembly language code, object code, or other instruction format that is interpreted by one or more processors. Some operations in method 1200 may be combined and/or the order of some operations may be changed.

[00147] Method 1200 is performed by a plurality computational machines that are included in a linear communication orbit. As described above, the linear communication orbit terminates at two end nodes that include a head node and a tail node. In some embodiments, the plurality of machines that are included in the linear communication orbit are interconnected (1202) by a single local area network. Additionally, in some embodiments, the linear communication orbit is organized (1204) from the plurality of machines based on a set of rules at a first time and subject to changes in configuration and membership when one or more machines join or exit the linear communication orbit.

[00148] A first machine included in the plurality of machines generates (1206) a data request to request specific data. In some implementations, the specific data comprises (1208) one of a plurality of shards of an object, such as a software application, software application update, database, or database update. Thus, the specific data is requested for the purpose of installing the software application or database, or updating the software application or database, in the first machine.

[00149] A plurality of the machines pass (1210) the data request along a data request path that tracks the linear communication orbit until the request is received at a second machine, in the linear communication orbit, that returns the specific data in response to the

data request. In some situations, the second machine comprises (1212) a machine, in the linear communication orbit, that has a cached copy of the specific data in its local cache. However, in other situations, when no machine in the request path has a cached copy of the specific data in its local cache, the second machine is (1212) the head node. In that circumstance, the head node sends the data request for the specific data to a server that is coupled to the head node, and receives a copy of the specific data returned by the server.

[00150] In some embodiments, the second machine is (1214) a machine closest, along the data request path, to the first machine that has a cached copy of the specific data in its local cache.

[00151] Additionally, in some implementations, passing the data request includes (1216) passing the data request along the data request path that tracks the linear communication orbit until (A) the request is received at a machine, in the linear communication orbit, that has a cached copy of the specific data in a local cache of the machine or (B) the request is received at the head node, which obtains and returns the specific data in response to the request.

[00152] After the data request is passed along the data request path, a third machine conditionally stores (1218) the specific data in a local cache of the third machine according to a data caching method, as described above. The third machine is located between the first machine and the second machine in the linear communication orbit.

[00153] In some embodiments, the data caching method includes (1220) a statistically random method of determining whether to cache respective data received from another machine. In some embodiments, the specific data comprises (1222) one of a plurality of shards in an object (i.e., where the plurality of shards together form, or can be combined to form, the object), the specific data is to be cached in the linear communication orbit, and the data caching method is (1222) based on at least one of the size of the specific data (e.g., the size of a particular shard), the size of the object (e.g., the total size of the plurality of shards that when combined form the object), the size of the local cache of the third machine, a total size of available caches in the linear communication orbit, and a total size of a data set that includes the object. More details and examples on the data caching method are explained above with reference to Figures 3A-3C and 11B (e.g., operations 1116, 1118).

[00154] In some circumstances, the data request generated by the first machine (e.g., at 1206) is a first data request, and the specific data is first specific data. Method 1200 includes (1224) additional operations to regenerate or reconstitute an object that is to be installed or updated at the first machine. In accordance with these additional operations, the first machine also generates (1226) a second data request to request second specific data. The second data request is passed (1228) along the data request path until the second data request is received at a fourth machine, in the linear communication orbit, that returns the second specific data in response to the second data request. In operation 1228, the fourth machine is distinct from the second machine. However, in some circumstances, the same second machine has locally cached copies of both the first specific data and second specific data, in which case the fourth machine is the same machine as the second machine. When the first machine receives the plurality of specific data, including the first and second specific data, it combines (1230) the plurality of specific data to generate the object for use at the first machine.

[00155] In some embodiments, the plurality of specific data comprise (1232) shards of an object specified in a manifest. For example, in some implementations, the manifest is received by the first machine with, or as a result of receiving, a system management message or data distribution command. The manifest includes a list of shards that, when combined, form the object to be installed. However, as described above, in some implementations or circumstances the manifest is a file or object obtained by the first machine in response to a request generated by the first machine. As described above, the first machine may generate the request for the manifest in response to a message received from a server (e.g., by evaluating a rule include in a server management message), in response to an event in an application executed by the first machine (e.g., an automated procedure that checks for application updates), in response to a user command, etc.

[00156] In some other implementations, the manifest is referenced by a command or system management message received by the first machine, and the first machine sends a data request to obtain a copy of the manifest, using the same methodology as described above with reference to Figures 11A-11D to obtain any other specific data. Upon receiving the manifest, the first machine proceeds to generate and transmit data requests for the shards listed in the manifest. In some embodiments, generating the first data request by the first

machine includes (1234) delaying generation of the first data request with respect to time at which the manifest is received by the first machine in accordance with a predefined request spreading method.

[00157] Under some circumstances, where the data request (see operation 1206) comprises a first data request, method 1200 includes (1236) additional operations to consolidate the first data request and a third data request when both request the same specific data. In accordance with these additional operations, a fifth machine generates (1238) the third data request to request the same specific data as the first request, after the first machine has generated the first data request. When predefined criteria in a request consolidation method are satisfied (e.g., where the first data request is still outstanding at the time the first machine receives the third data request, and both request the same specific data), the plurality of machines in the linear communication orbit forgo (1240) passing the third data request along at least a part of a respective data request path that is associated with the third data request. In these circumstances, the third data request is resolved (1242) by conveying the specific data from the second machine to the first machine, and then the first machine conveying the specific data (from the first machine) to the fifth machine.

[00158] Additionally, in some embodiments, method 1200 further comprises passing (1244) the specific data from the second machine to the first machine through a plurality of intervening machines in the linear communication orbit. Each machine of the plurality of intervening machines, located between the second machine and the first machine, caches (1246) a copy of the specific data in a local cache of the machine if (i.e., in accordance with a determination that) predefined local caching criteria are met. Stated another way, a copy of the specific data is locally cached at a subset of the intervening machines, and each intervening machine independently determines whether it should locally cache a copy of the specific data in accordance with the predefined local caching criteria.

[00159] However, in some situations, the specific data not only are passed (1248) from the second machine to the first machine through a plurality of intervening machines in the linear communication orbit, but also continue (1248) to be passed to additional machines in the linear communication orbit (e.g., machines upstream or downstream of the first machine), until the specific data are received by one of the end nodes, i.e., the head node or the tail node. A plurality of machines are selected from the plurality of intervening machines and the

additional machines. Each of the plurality of machines caches (1250) a copy of the specific data in a local cache of the machine in accordance with a determination that predefined local caching criteria are met. In those situations in which the second machine is (1252) the tail node or the head node, the specific data are returned from the second machine to the first machine and conditionally stored in the plurality of machines, including the third machine (see discussion above re 1218-1222), that are coupled within the linear communication orbit and do not have the specific data (i.e., do not already have a locally cached copy of the specific data).

[00160] In one specific example, the second machine is (1254) a downstream machine between the first machine and the tail node. The data request is passed downstream from the first machine to the second machine, and the specific data are returned upstream from the second machine to the first machine. Similarly, in another specific example, the second machine is (1256) an upstream machine between the first machine and the head node, and the data request is passed from the first machine downstream to the tail node and then upstream to the second machine, and the specific data are returned downstream from the second machine to the first machine.

[00161] It should be understood that the particular order in which the operations in Figures 12A-12G have been described are merely exemplary and are not intended to indicate that the described order is the only order in which the operations could be performed. One of ordinary skill in the art would recognize various ways to reorder the operations described herein. Additionally, it should be noted that details of other processes described herein with respect to method 1100 (e.g., Figures 11A-11D) are also applicable in an analogous manner to method 1100 described above with respect to Figures 12A-12G. For brevity, these details are not repeated here.

[00162] The foregoing description has been provided with reference to specific embodiments. However, the illustrative discussions above are not intended to be exhaustive or to be limiting to the precise forms disclosed. Many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles disclosed and their practical applications, to thereby enable others to best utilize the disclosure and various embodiments with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A method of data caching and distribution, comprising:
at a computational machine having a local cache:
receiving from a first machine a data request that is used to request specific data;
determining whether the computational machine stores the specific data in the local cache; and
in accordance with a determination that the computational machine does not store the specific data in the local cache, responding to the data request, by:
passing the data request to a second machine, and
when the second machine returns the specific data to the computational machine, returning the specific data to the first machine and determining whether to store the specific data in the local cache according to a data caching method.
2. The method of claim 1, further comprising:
in accordance with a determination that the computational machine stores the specific data in its local cache, responding to the data request by returning the specific data to the first machine.
3. The method of any of claims 1-2, wherein:
the data caching method comprises a statistically random method of determining whether to cache respective data received from another machine.
4. The method of any preceding claim, wherein the data caching method is based on at least one of the size of the specific data, the size of the local cache, the size of available caches in a linear communication orbit that includes the computational machine, and the size of a respective data set that includes the specific data.
5. The method of any preceding claim, wherein the specific data comprises one of a plurality of shards of an object, wherein the object comprises a data file, software application, software application update, software patch, database, or database update.

6. The method of any preceding claim, wherein the computational machine is a head node of a linear communication orbit that is organized from a plurality of machines, including the computational machine and the first machine, based on a set of rules at a specific time, and the second machine is a server coupled to the head node.
7. The method of any preceding claim, wherein the data request comprises a first data request, the method further comprising receiving at the computational machine, after receiving the first data request, a second data request to request the same specific data as the first data request, and in accordance with a determination that predefined criteria are satisfied, forgoing passing the second data request to the second machine.
8. The method of any of claims 1-5 and 7, wherein the computational machine is an intermediate machine between neighboring machines within a linear communication orbit, the linear communication orbit being organized from a plurality of machines based on a set of rules at a specific time and subject to changes in configuration and membership when one or more machines join or exit the linear communication orbit.
9. The method of claim 8, wherein the computational machine is downstream of the first machine and upstream of the second machine in the linear communication orbit.
10. The method of claim 8, wherein the computational machine is downstream of the second machine and upstream of the first machine in the linear communication orbit.
11. The method of any preceding claim, wherein the computational machine and the first machine are interconnected by a single local area network.
12. A method of data caching and distribution, comprising:
 - at a computational machine having a local cache:
 - generating a first data request that is used to request specific data;
 - passing the data request to a second machine;
 - receiving at the computational machine, from a third machine, a second data request to request the specific data after generating the first data request, and in accordance with a determination that predefined criteria are satisfied, forgoing passing the second data request to the second machine; and

when the second machine returns the specific data to the computational machine, returning the specific data to the third machine.

13. A non-transitory computer-readable medium, having instructions stored thereon, which when executed by one or more processors cause the processors to perform operations comprising:

at a computational machine that has a local cache:

receiving from a first machine a data request that is used to request specific data;

determining whether the computational machine stores the specific data in the local cache; and

in accordance with a determination that the computational machine does not store the specific data in the local cache, responding to the data request, by

passing the data request to a second machine, and

when the second machine returns the specific data to the computational machine, returning the specific data to the first machine and determining whether to store the specific data in the local cache according to a data caching method.

14. The non-transitory computer-readable medium of claim 13, wherein the instructions stored in the memory include instructions for causing the computational machine to perform the method of any of claims 2-12.

15. The non-transitory computer-readable medium of claim 13, wherein the instructions, when executed by the one or more processors cause the one or more processors to perform further operations comprising:

in accordance with a determination that the computational machine stores the specific data in its local cache, responding to the data request by returning the specific data to the first machine.

16. The non-transitory computer-readable medium of claim 13, wherein the data caching method comprises a statistically random method of determining whether to cache respective data received from another machine.

17. The non-transitory computer-readable medium of claim 13, wherein the data caching method is based on at least one of the size of the specific data, the size of the local cache, the size of available caches in a linear communication orbit that includes the computational machine, and the size of a respective data set that includes the specific data.

18. The non-transitory computer-readable medium of claim 13, wherein the specific data comprises one of a plurality of shards of an object, wherein the object comprises a data file, software application, software application update, software patch, database, or database update.

19. The non-transitory computer-readable medium of claim 13, wherein the computational machine is a head node of a linear communication orbit that is organized from a plurality of machines, including the computational machine and the first machine, based on a set of rules at a specific time, and the second machine is a server coupled to the head node.

20. A computational machine, comprising:

one or more processors;

a local cache; and

memory having instructions stored thereon, which when executed by the one or more processors cause the processors to perform operations, comprising:

at the computational machine:

receiving from a first machine a data request that is used to request specific data;

determining whether the computational machine stores the specific data in the local cache; and

in accordance with a determination that the computational machine does not store the specific data in the local cache, responding to the data request, by

passing the data request to a second machine, and

when the second machine returns the specific data to the computational machine, returning the specific data to the first machine and determining whether to store the specific data in the local cache according to a data caching method.

21. The computational machine of claim 20, wherein the instructions stored in the memory include instructions for causing the computational machine to perform the method of any of claims 2-12.
22. A computational machine, comprising:
one or more processors;
a local cache; and
means for receiving from a first machine a data request that is used to request specific data;
means for determining whether the computational machine stores the specific data in the local cache; and
means for responding to the data request, in accordance with a determination that the computational machine does not store the specific data in the local cache, by
passing the data request to a second machine, and
when the second machine returns the specific data to the computational machine, returning the specific data to the first machine and determining whether to store the specific data in the local cache according to a data caching method.
23. The computational machine of claim 22, including means for causing the computational machine to perform the method of any of claims 2-12.
24. A method of data caching and distribution, comprising:
at a plurality of machines that are included in a linear communication orbit, wherein the linear communication orbit includes two end nodes, comprising a head node and a tail node:
generating a data request by a first machine to request specific data;
passing the data request along a data request path that tracks the linear communication orbit until the request is received at a second machine, in the linear communication orbit, that returns the specific data in response to the data request; and
at a third machine between the second machine and the first machine in the linear communication orbit conditionally storing the specific data in a local cache of the third machine according to a data caching method.

25. The method of claim 24, wherein the second machine comprises a machine, in the linear communication orbit, that has a cached copy of the specific data in its local cache or the head node if no machine in the request path has a cached copy of the specific data in its local cache.
26. The method of claim 24, wherein the second machine comprises a machine closest, along the data request path, to the first machine that has a cached copy of the specific data in its local cache.
27. The method of claim 24, wherein passing the data request includes passing the data request along the data request path that tracks the linear communication orbit until (A) the request is received at a machine, in the linear communication orbit, that has a cached copy of the specific data in a local cache of the machine or (B) the request is received at the head node, which obtains and returns the specific data in response to the request.
28. The method of any one of claims 24-27, wherein the plurality of machines that are included in the linear communication orbit are interconnected by a single local area network.
29. The method of any one of claims 24-28, wherein:
the data caching method comprises a statistically random method of determining whether to cache respective data received from another machine.
30. The method of any one of claims 24-29, wherein the specific data comprises one of a plurality of shards, in an object, to be cached in the linear communication orbit, and wherein the data caching method is based on at least one of the size of the specific data, the size of the object, the size of the local cache of the third machine, a total size of available caches in the linear communication orbit, and a total size of a data set that includes the object.
31. The method of any one of claims 24-30, wherein the data request comprises a first data request and the specific data comprises first specific data, the method including:
generating, at the first machine, a second data request to request second specific data;
passing the second data request along the data request path until the request is received at a fourth machine, in the linear communication orbit, that returns the second specific data in response to the second data request, wherein the fourth machine is distinct from the second machine; and

combining, at the first machine, a plurality of specific data, including the first specific data and the second specific data, to generate an object for use at the first machine.

32. The method of claim 31, wherein the plurality of specific data comprise shards of said object specified in a manifest.

33. The method of claim 32, wherein generating the first data request by the first machine includes delaying generation of the first data request, with respect to time at which the manifest is received by the first machine, in accordance with a predefined request spreading method.

34. The method of any one of claims 24-33, wherein the data request comprises a first data request, the method further comprising:

generating at a fifth machine a third data request to request the specific data after generating the first data request by the first machine; and

in accordance with a determination that predefined criteria in a request consolidation method are satisfied, forgoing passing the third data request along at least a part of a respective data request path that is associated with the third data request.

35. The method of claim 33, wherein the specific data are conveyed to the fifth machine by returning the specific data to the first machine, which reconveys the specific data to the fifth machine.

36. The method of any one of claims 24-35, wherein the specific data comprise one of a plurality of shards of an object, such as a software application, software application update, database, or database update.

37. The method of any one of claims 24-35, further comprising:

passing the specific data from the second machine to the first machine through a plurality of intervening machines in the linear communication orbit;

at each machine of the plurality of intervening machines, located between the second machine and the first machine, caching a copy of the specific data in a local cache of the machine in accordance with a determination that predefined local caching criteria are met.

38. The method of any one of claims 24-35, further comprising:

passing the specific data from the second machine to the first machine through a plurality of intervening machines in the linear communication orbit and continuing to pass the specific data to additional machines in the linear communication orbit until the specific data are received by one of the end nodes; and

at each machine of a plurality of machines, selected from the plurality of intervening machines and the additional machines, caching a copy of the specific data in a local cache of the machine in accordance with a determination that predefined local caching criteria are met.

39. The method of claim 37 or claim 38, wherein the second machine is a downstream machine between the first machine and the tail node, the data request being passed downstream from the first machine to the second machine, the specific data being returned upstream from the second machine to the first machine.

40. The method of claim 37 or claim 38, wherein the second machine is an upstream machine between the first machine and the head node, the data request being passed from the first machine downstream to the tail node and upstream to the second machine, the specific data being returned downstream from the second machine to the first machine.

41. The method of claim 38, wherein the second machine is the tail node or the head node, the specific data being returned from the second machine to the first machine and conditionally stored in the plurality of machines, including the third machine, that are coupled within the linear communication orbit and do not have the specific data.

42. The method of any one of claims 24-41, wherein the linear communication orbit is organized from the plurality of machines based on a set of rules at a first time and subject to changes in configuration and membership when one or more machines join or exit the linear communication orbit.

43. A system, comprising:

one or more processors included in a plurality of machines;

memory having instructions stored thereon, which when executed by the one or more processors cause the processors to perform operations comprising:

at a plurality of machines that are included in a linear communication orbit, wherein the linear communication orbit terminates at two end nodes, comprising a head node and a tail node:

generating a data request by a first machine to request specific data;
passing the data request along a data request path that tracks the linear communication orbit until the request is received at a second machine, in the linear communication orbit, that returns the specific data in response to the data request; and
at a third machine between the second machine and the first machine in the linear communication orbit conditionally storing the specific data in a local cache of the third machine according to a data caching method.

44. The system of claim 43, wherein the instructions stored in the memory include instructions for causing the plurality of computational machines to perform the method of any of claims 24-41.

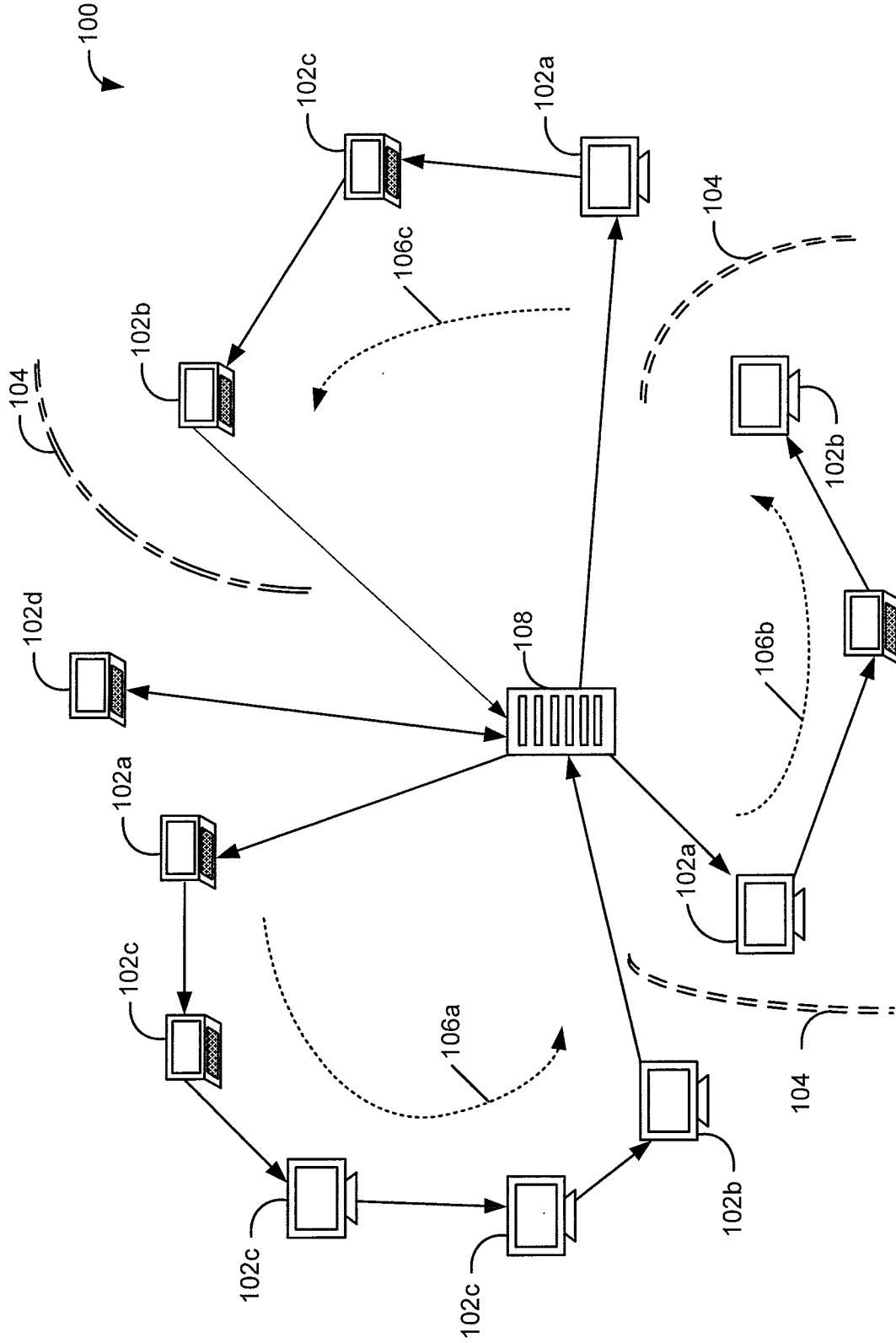


Figure 1A

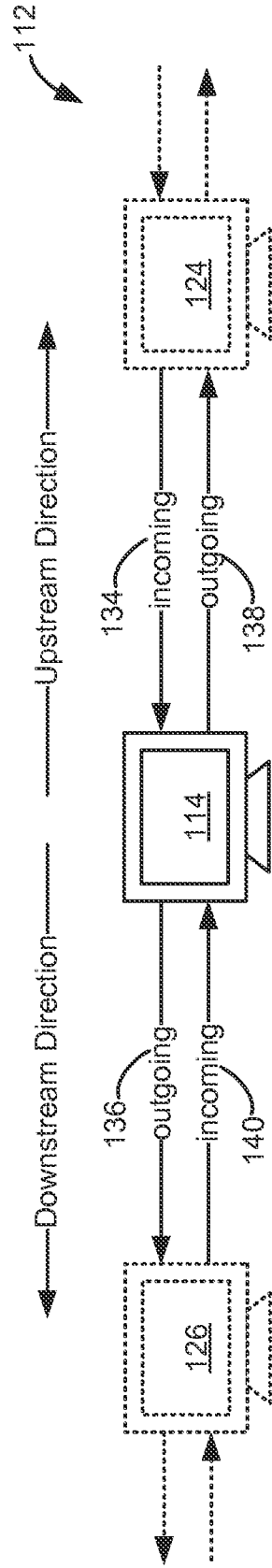


Figure 1B

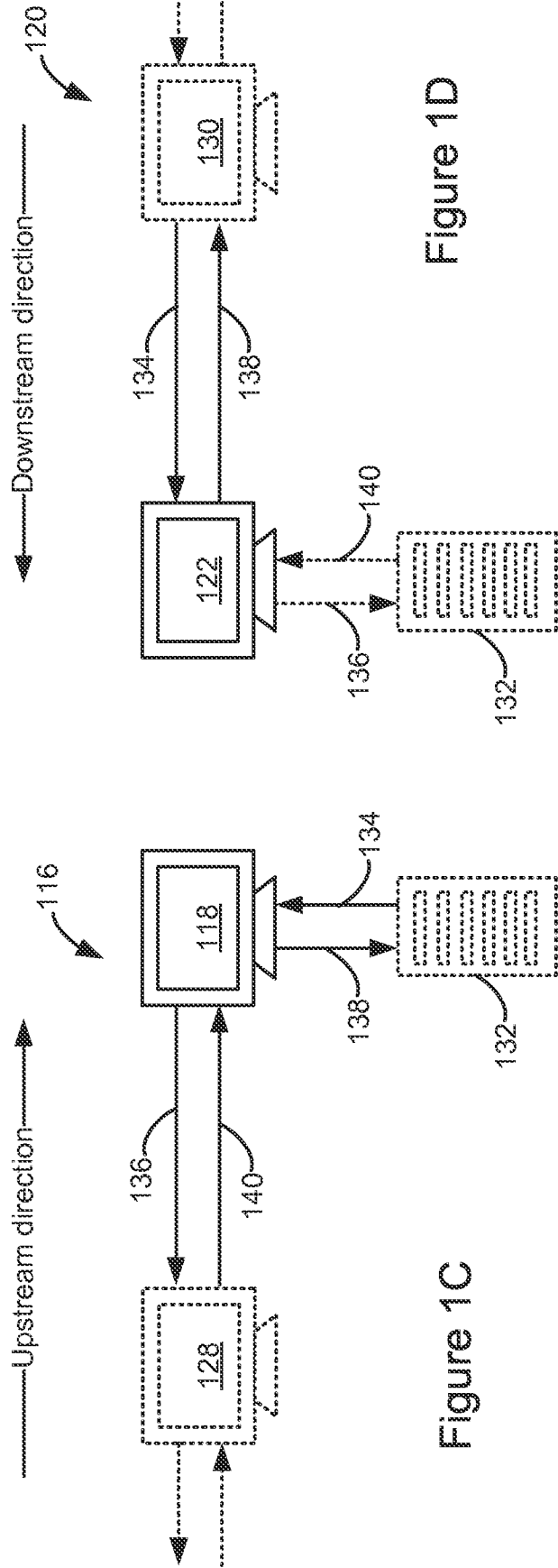


Figure 1C

Figure 1D

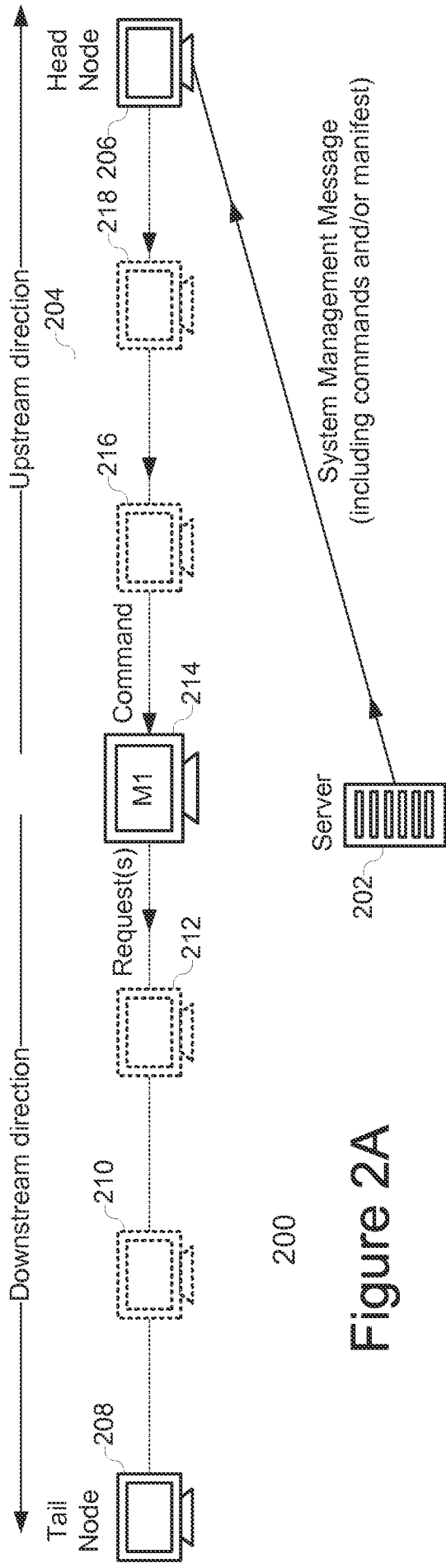


Figure 2A

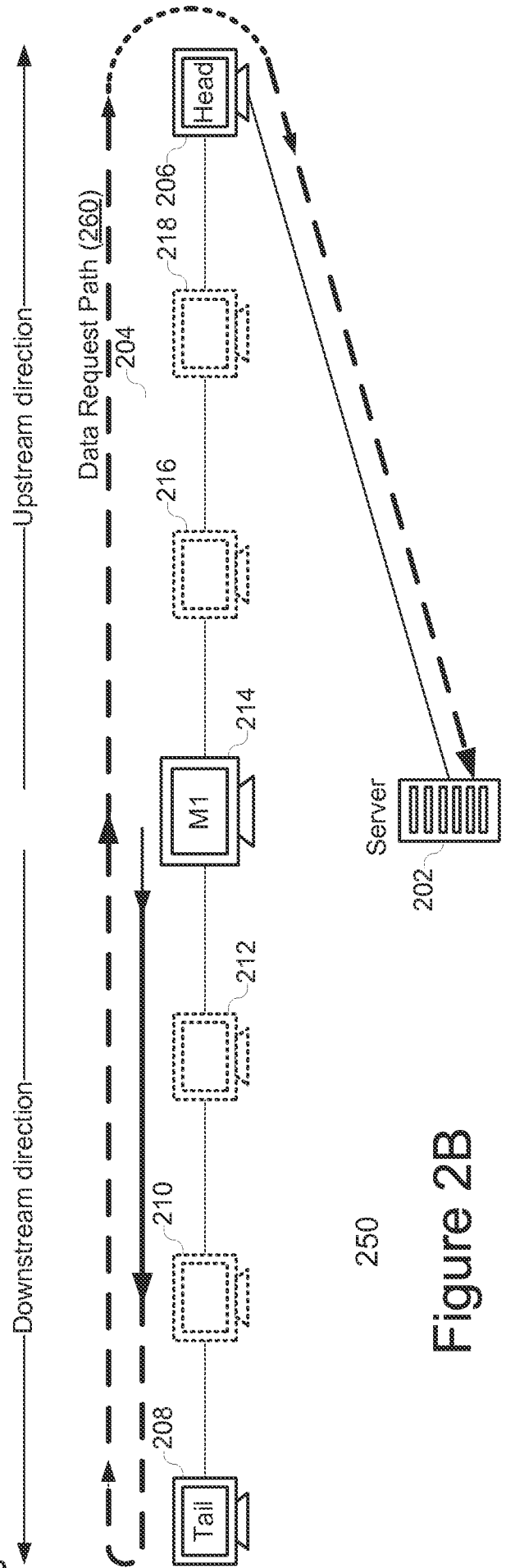


Figure 2B

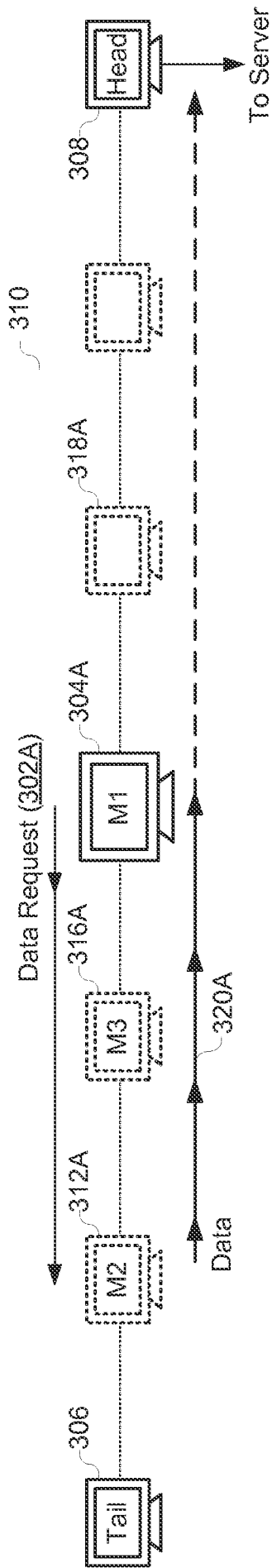


Figure 3A

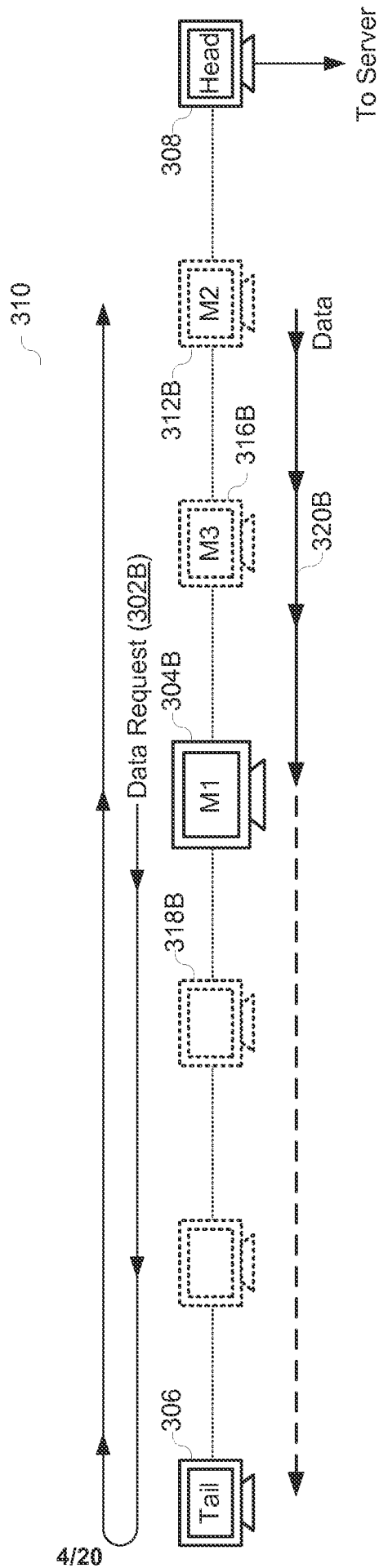


Figure 3B

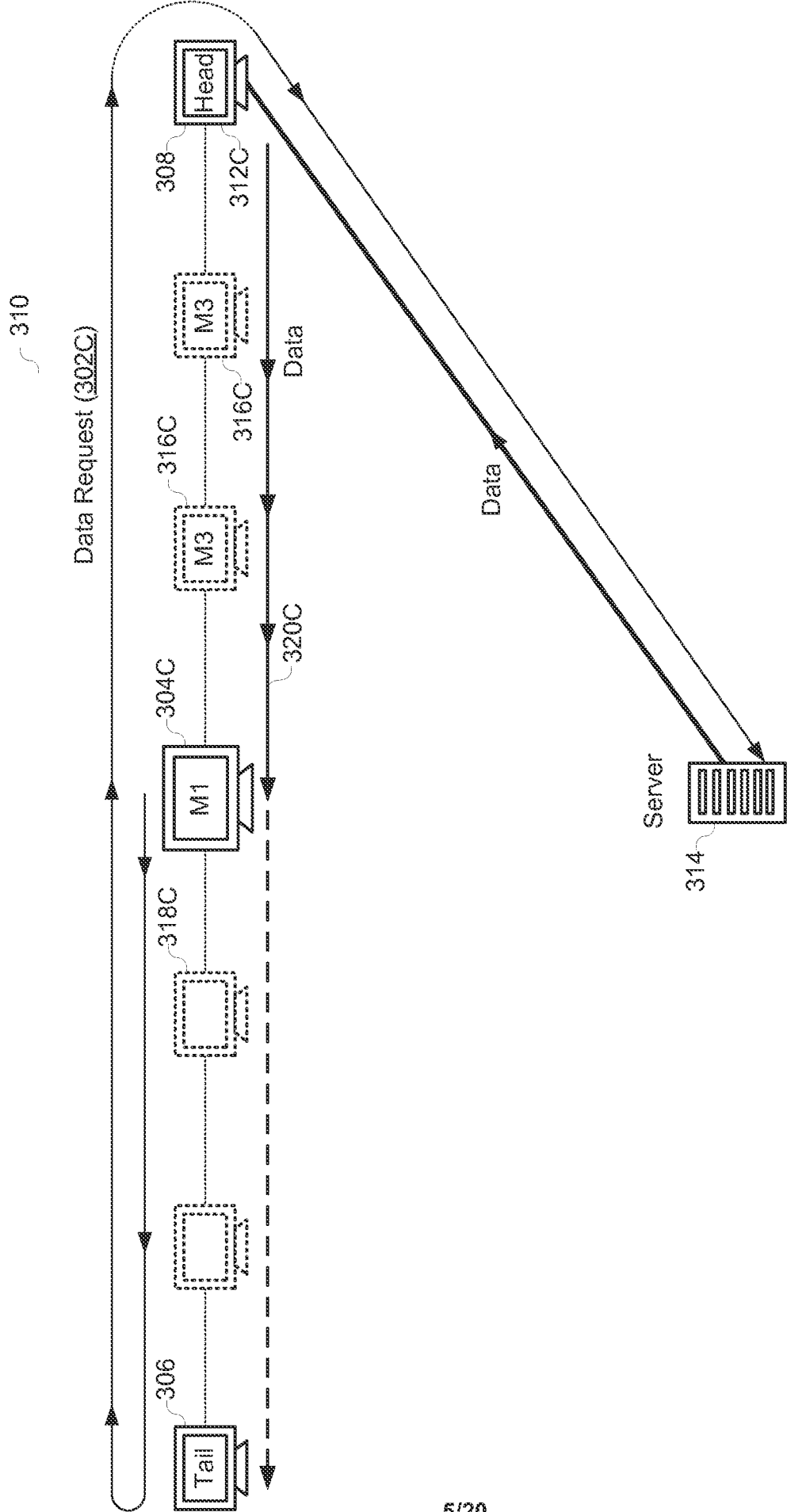


Figure 3C

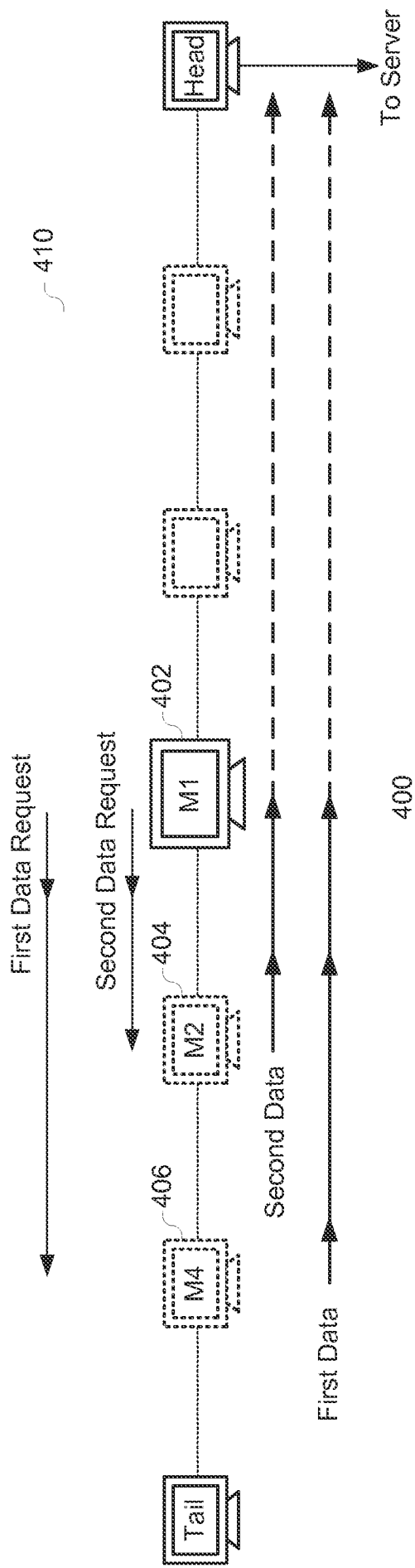


Figure 4

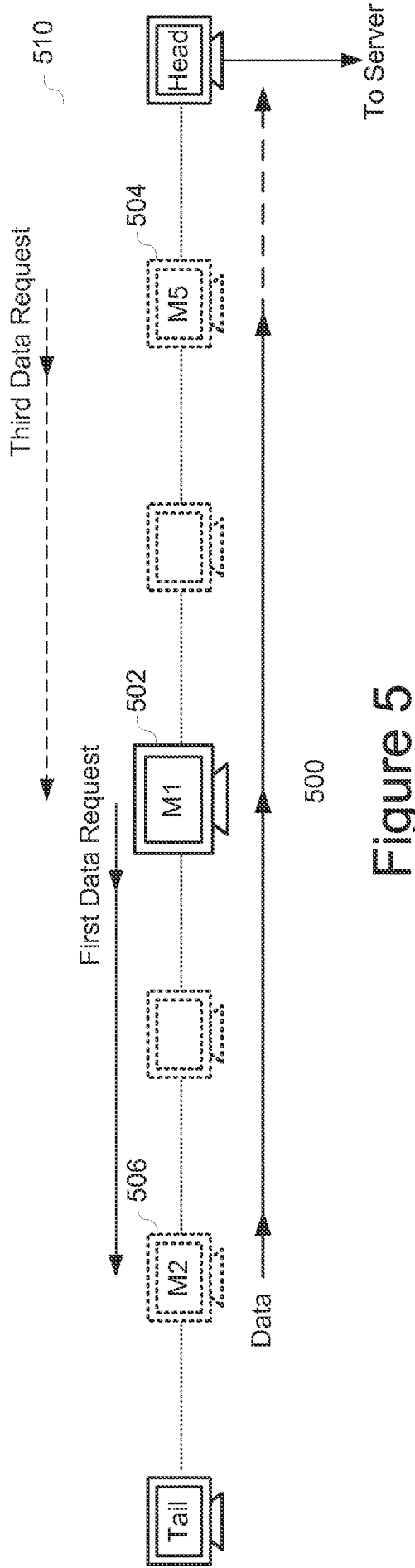
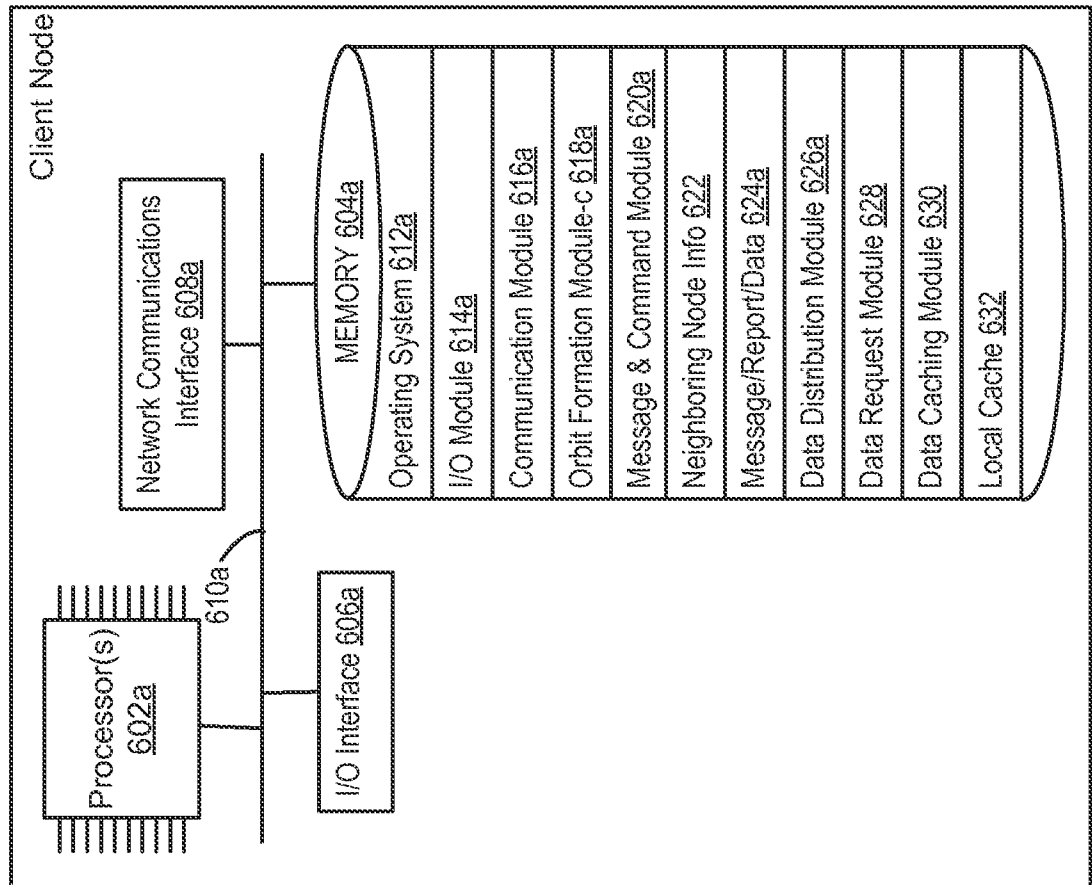
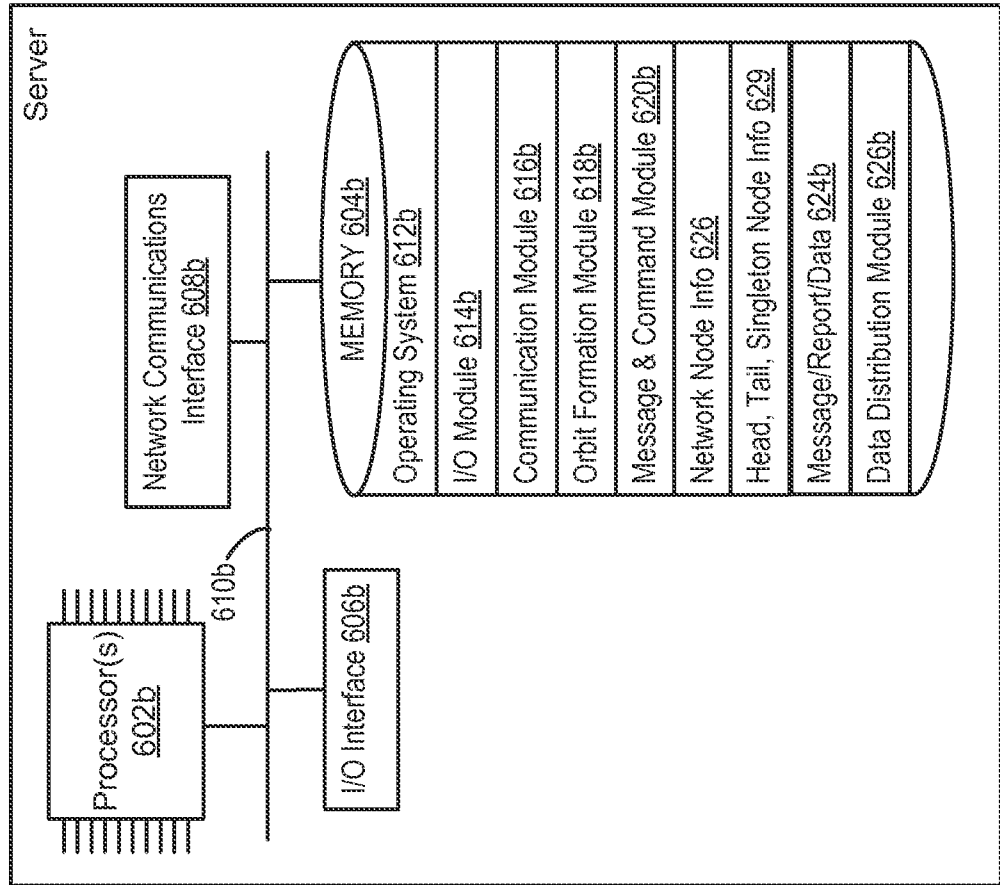


Figure 5

600 → **Figure 6A**



650 → **Figure 6B**



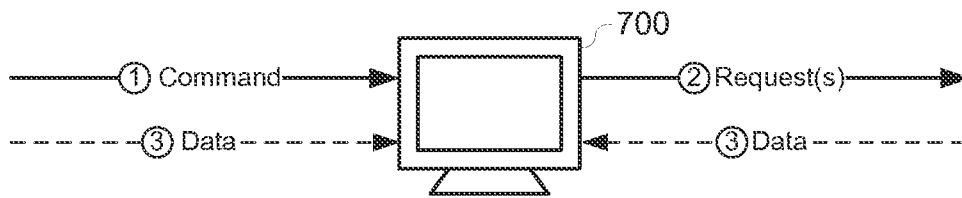


Figure 7

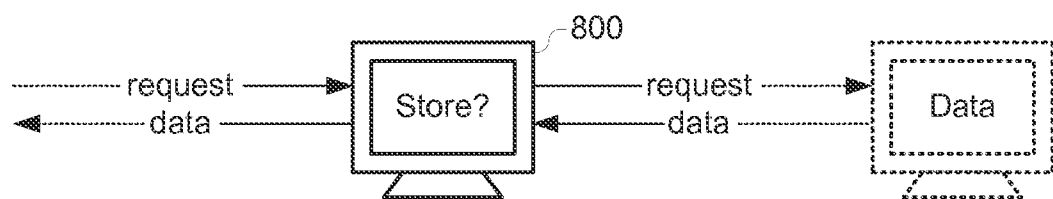


Figure 8A

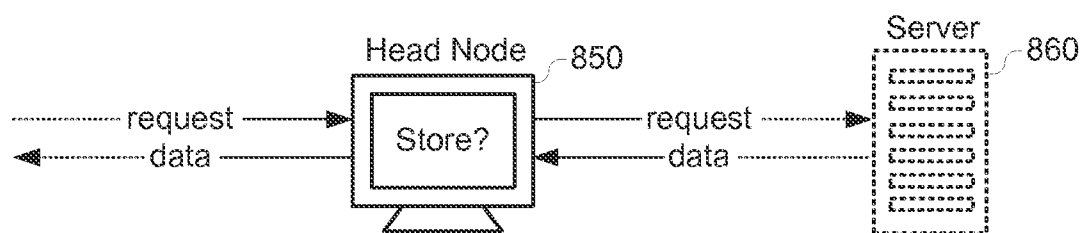


Figure 8B

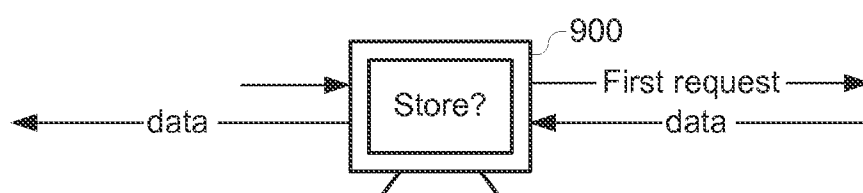


Figure 9

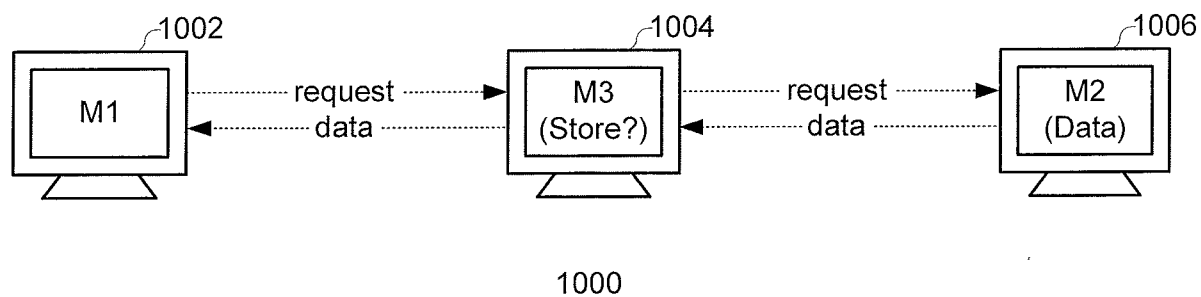


Figure 10A

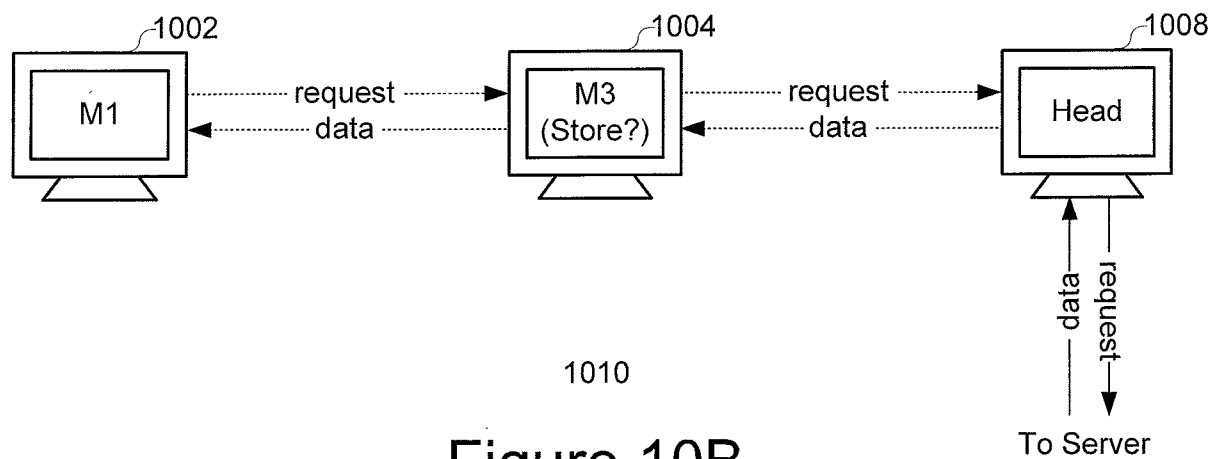
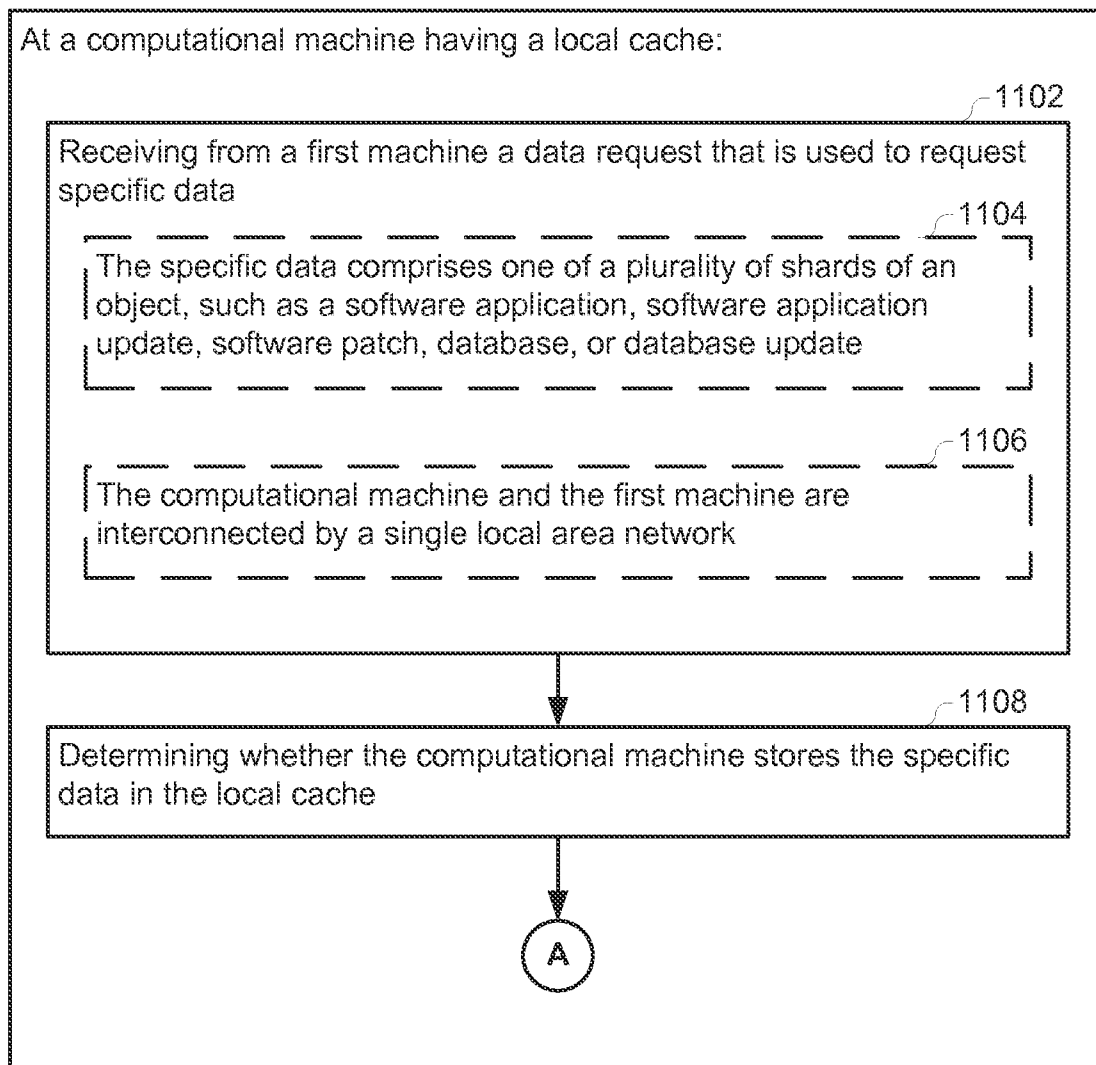


Figure 10B

1100**Figure 11A**

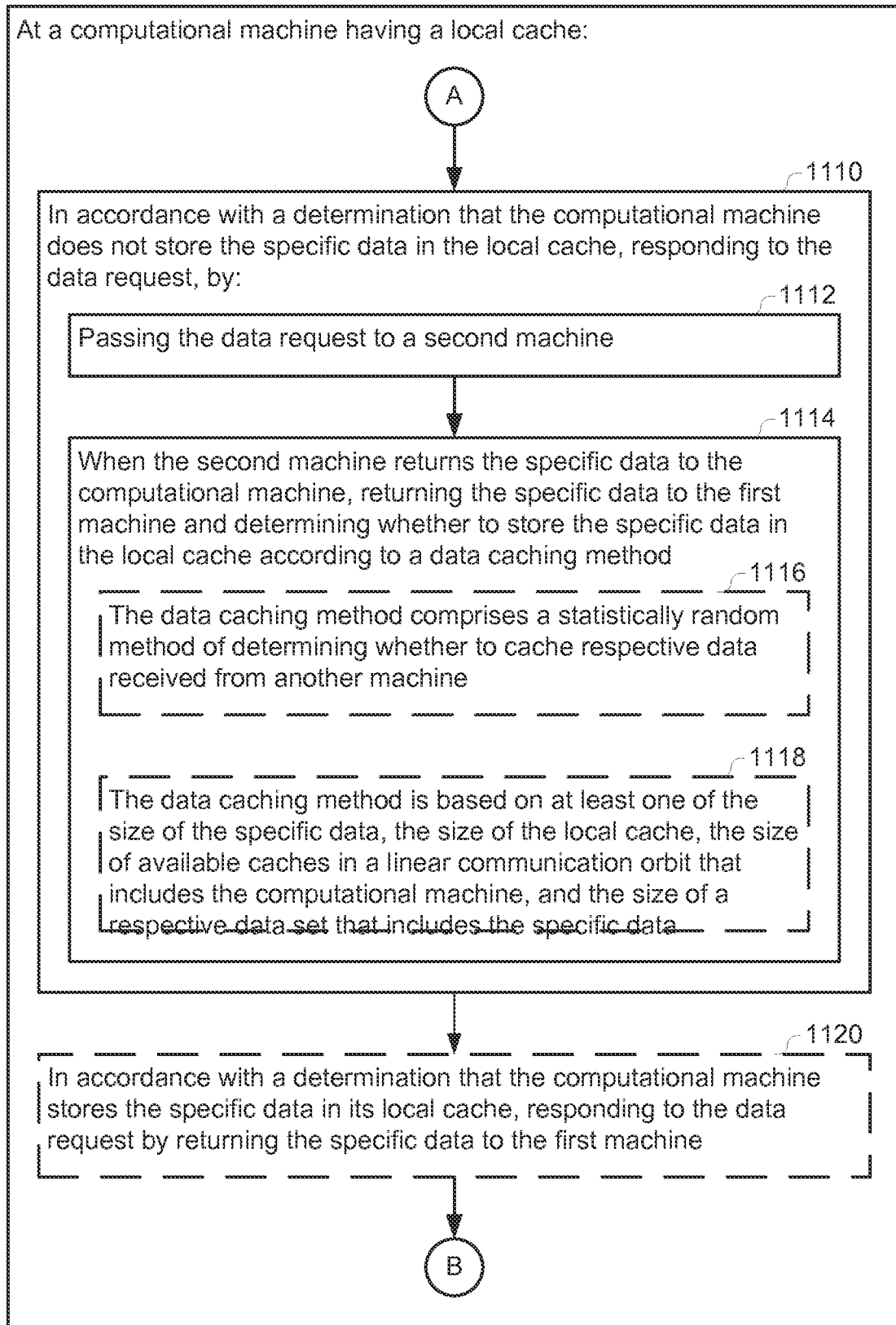


Figure 11B

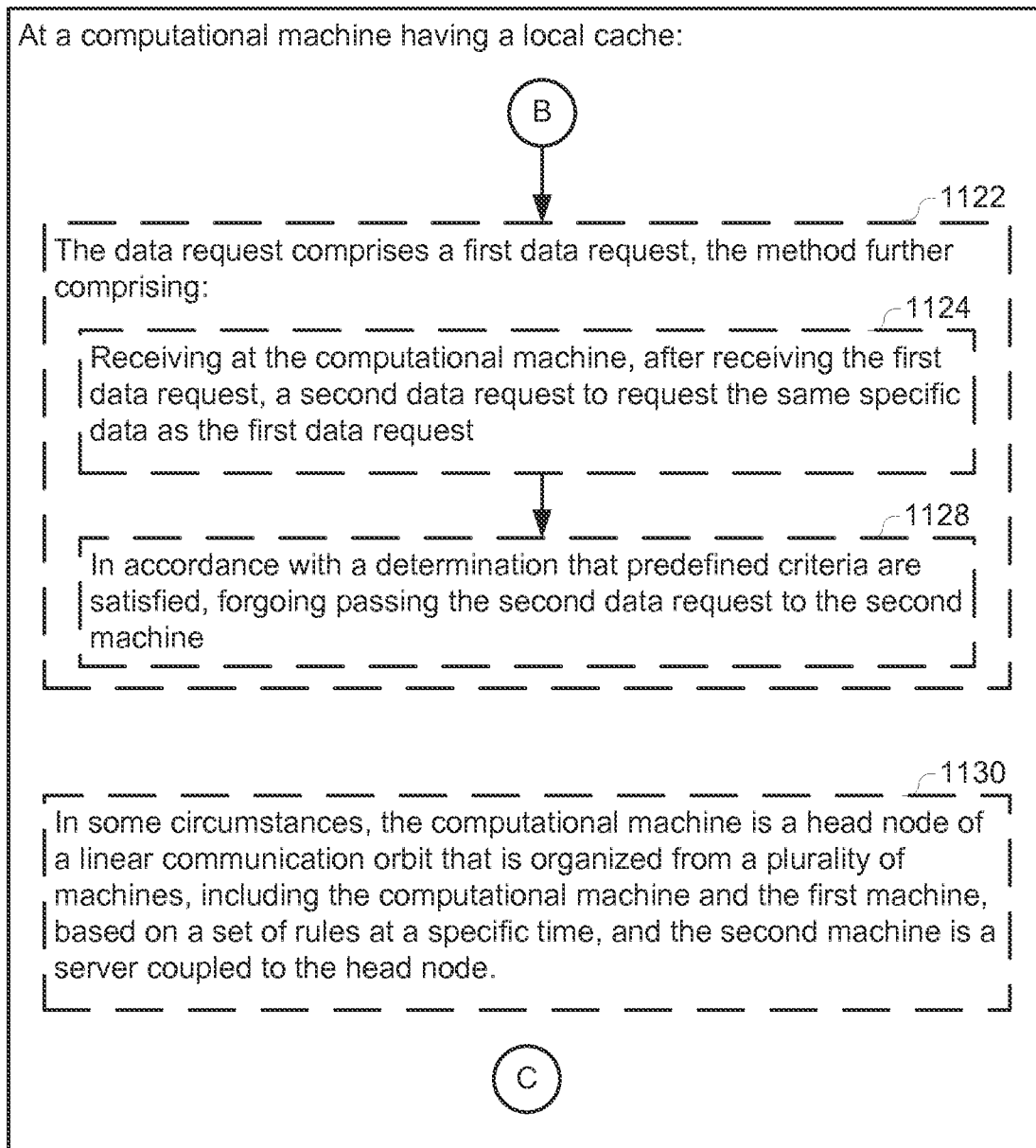


Figure 11C

At a computational machine having a local cache:



1132

In some circumstances, the computational machine is an intermediate machine between neighboring machines within a linear communication orbit, the linear communication orbit being organized from a plurality of machines based on a set of rules at a specific time and subject to changes in configuration and membership when one or more machines join or exit the linear communication orbit.

1134

In some circumstances, the computational machine is downstream of the first machine and upstream of the second machine in the linear communication orbit

1136

Alternatively, the computational machine is downstream of the second machine and upstream of the first machine in the linear communication orbit

Figure 11D

At a plurality of machines that are included in a linear communication orbit, wherein the linear communication orbit terminates at two end nodes, comprising a head node and a tail node:

1202
The plurality of machines that are included in the linear communication orbit are interconnected by a single local area network

1204
The linear communication orbit is organized from the plurality of machines based on a set of rules at a first time and subject to changes in configuration and membership when one or more machines join or exit the linear communication orbit

1206
Generating a data request by a first machine to request specific data

1208
The specific data comprise one of a plurality of shards of an object, such as a software application, software application update, database, or database update

A

1200

Figure 12A

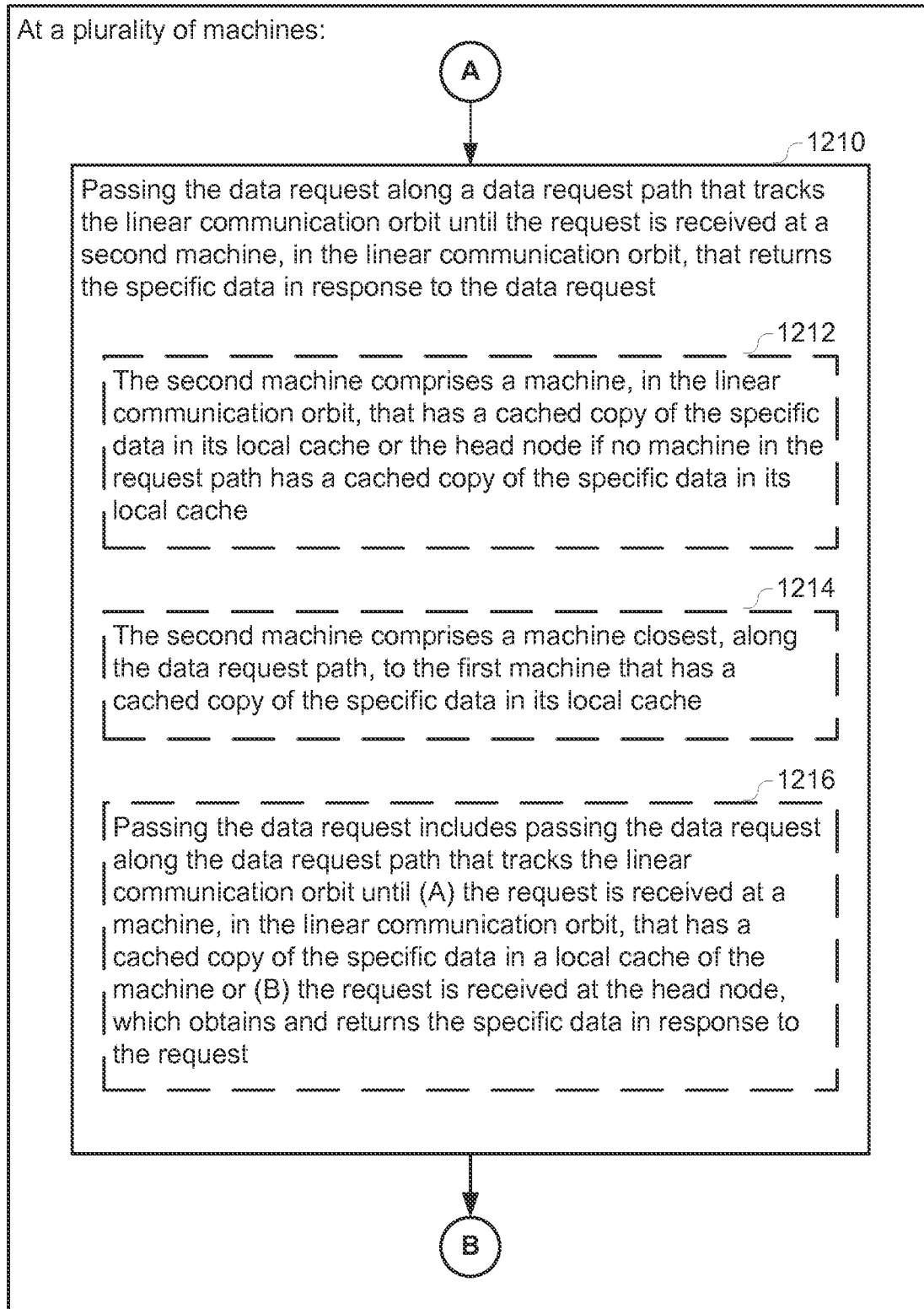


Figure 12B

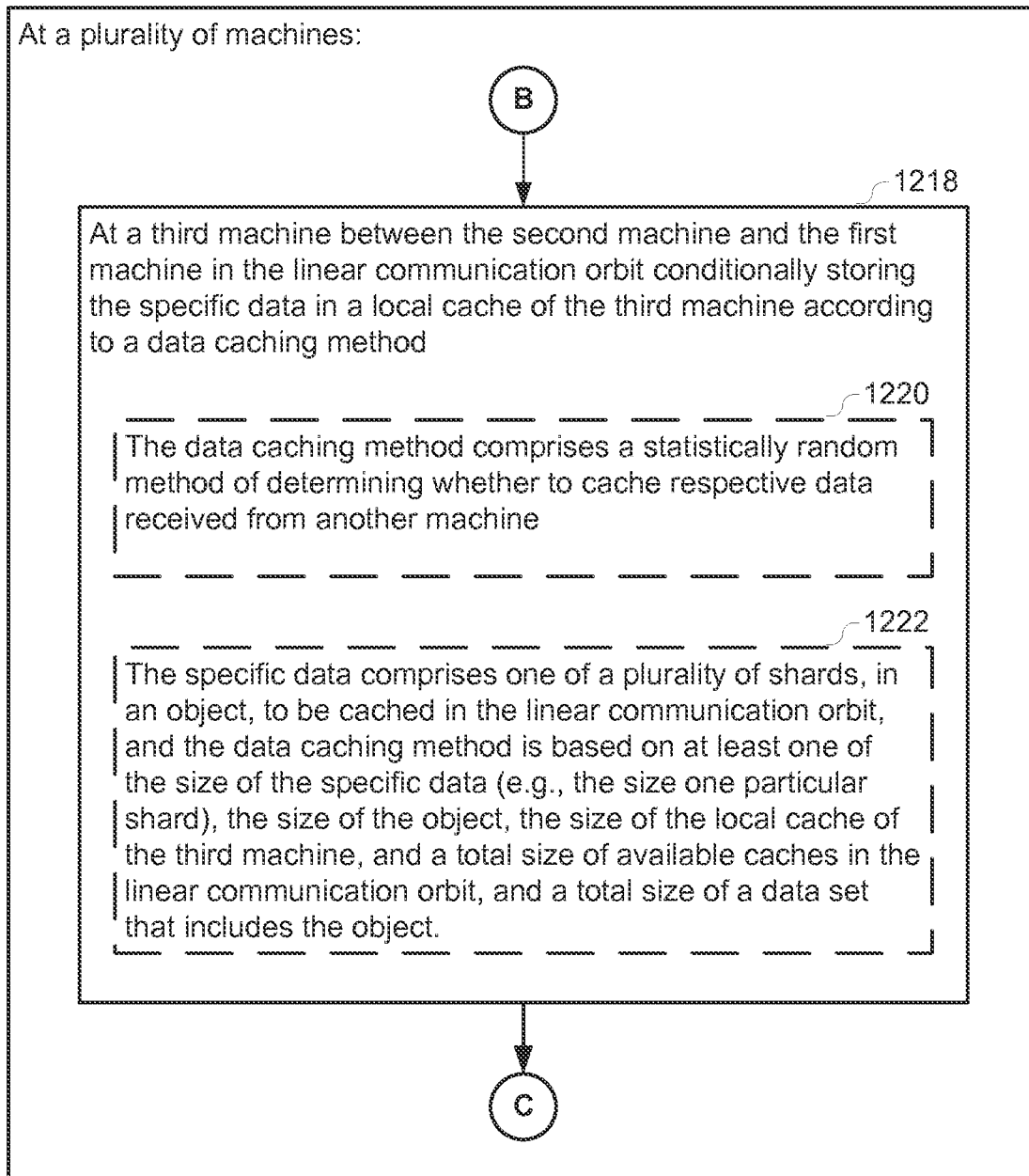


Figure 12C

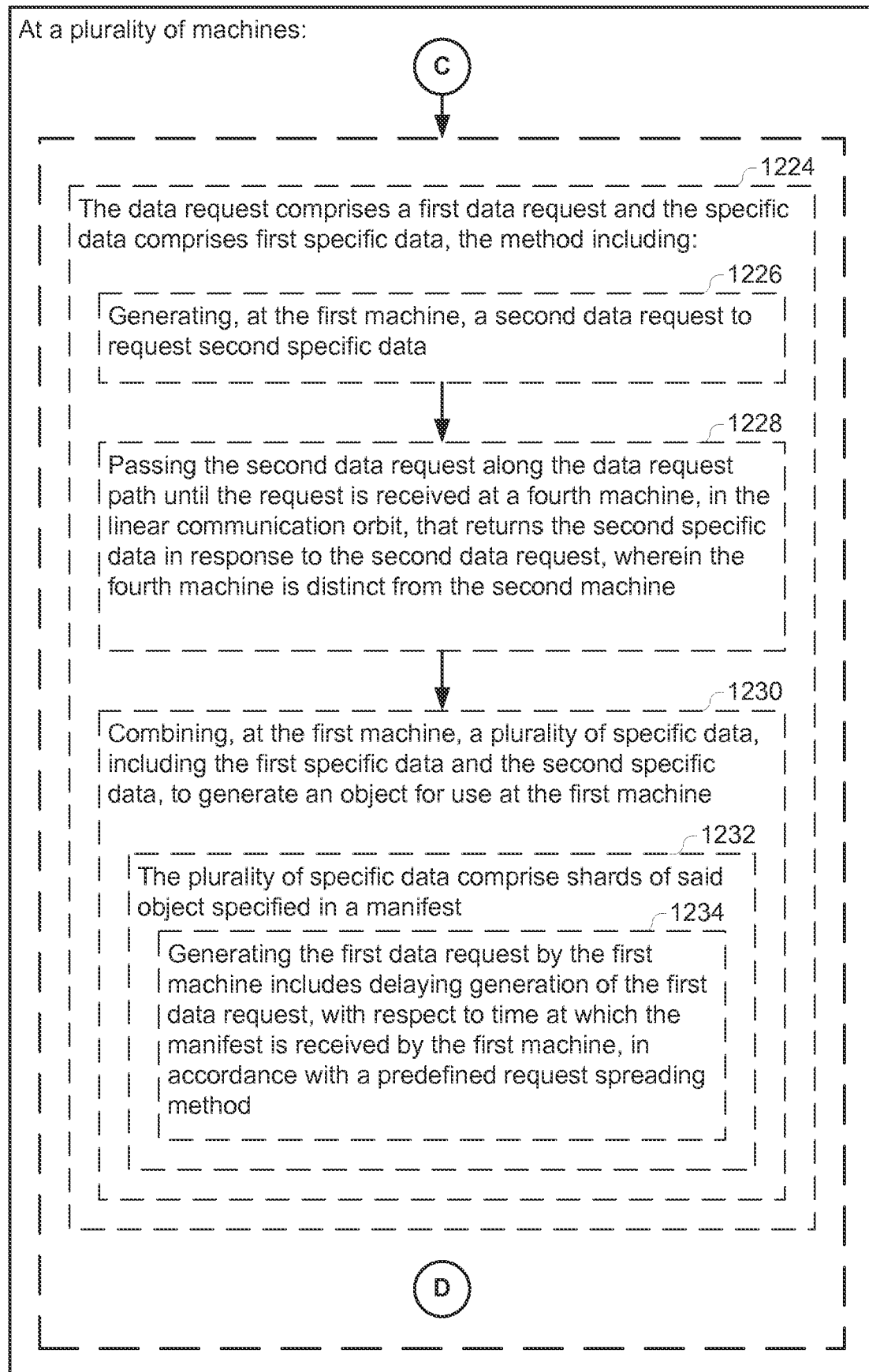


Figure 12D

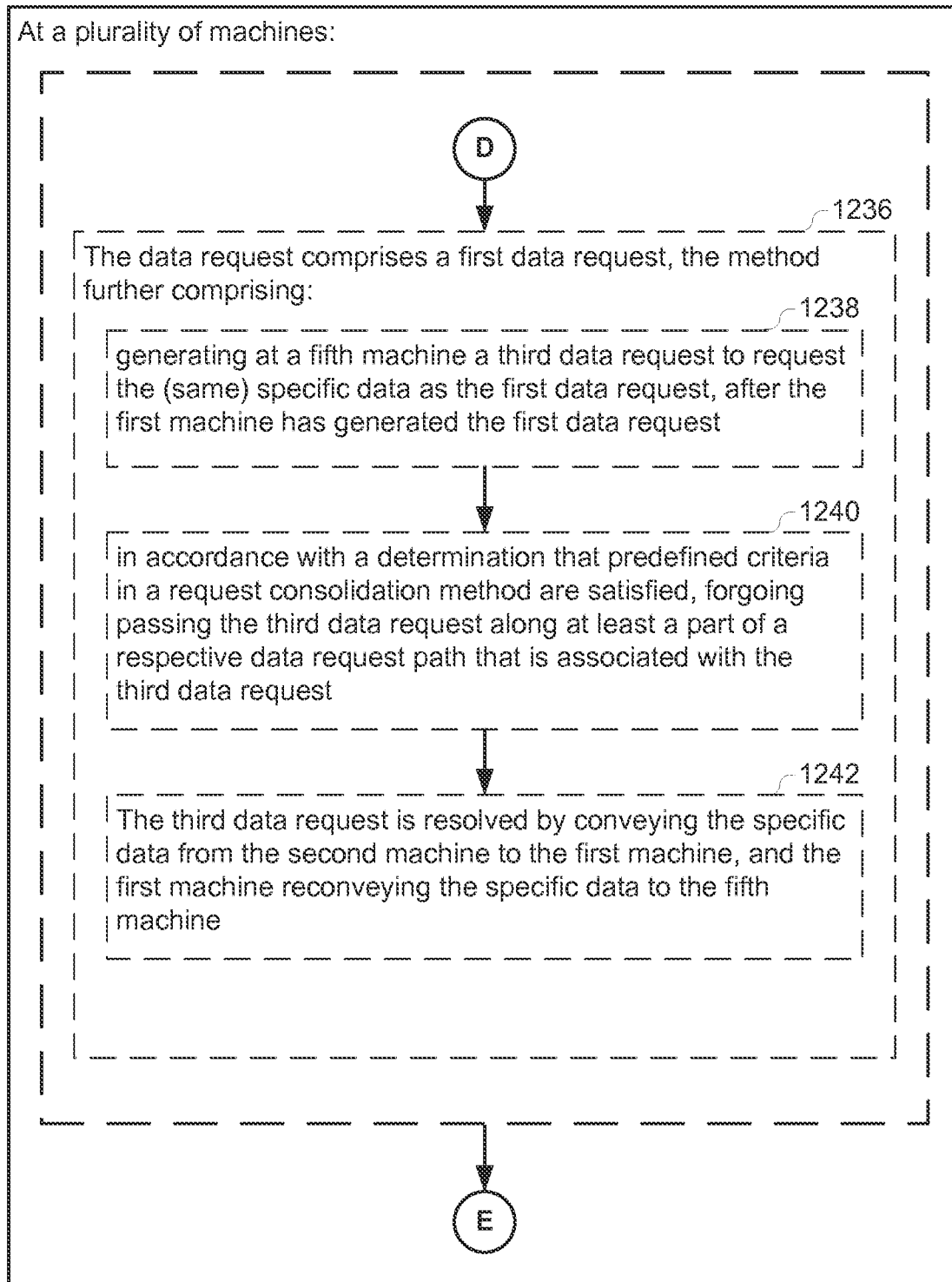


Figure 12E

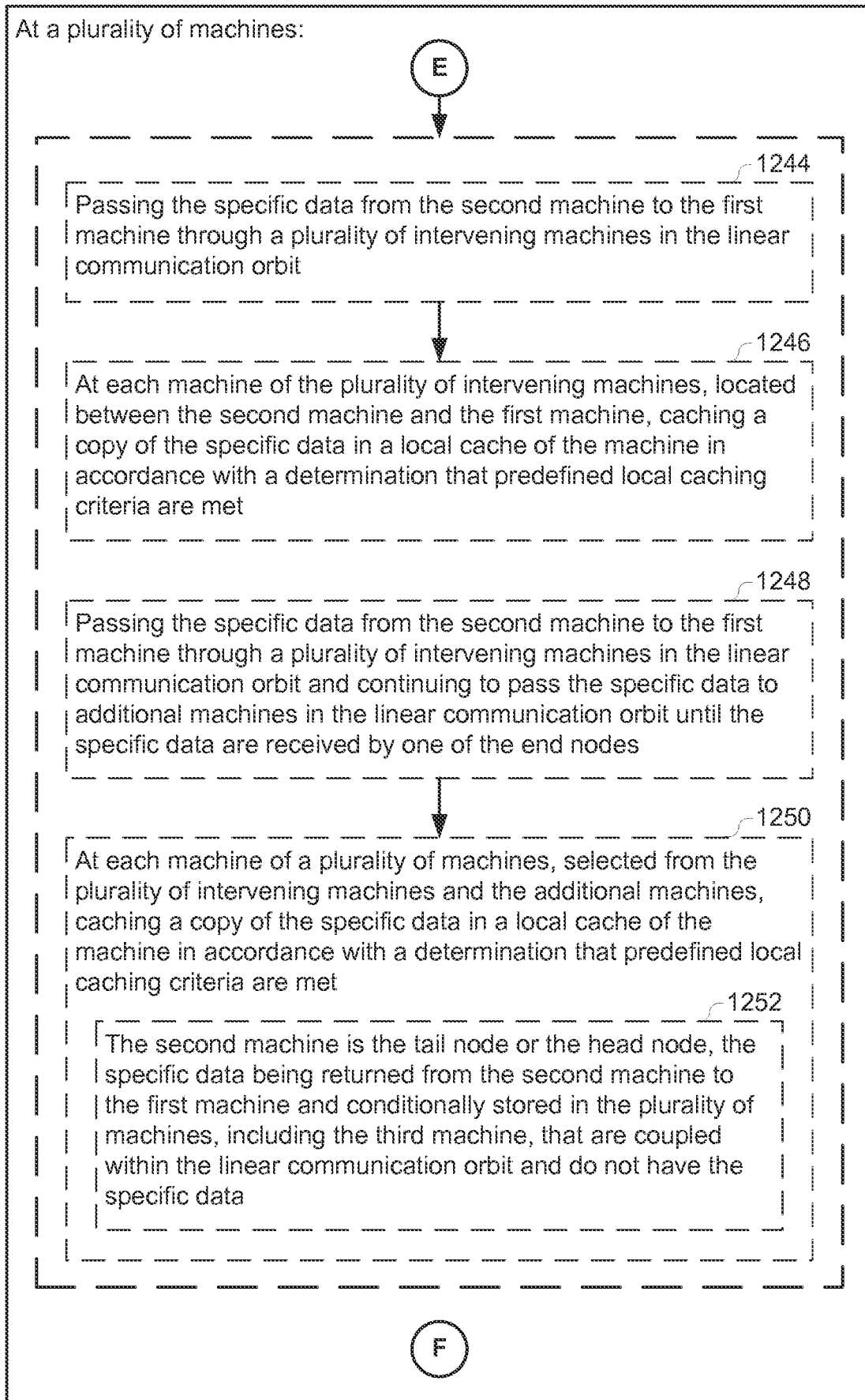


Figure 12F

At a plurality of machines:

F

1254

The second machine is a downstream machine between the first machine and the tail node, the data request being passed downstream from the first machine to the second machine, the specific data being returned upstream from the second machine to the first machine

1256

The second machine is an upstream machine between the first machine and the head node, the data request being passed from the first machine downstream to the tail node and upstream to the second machine, the specific data being returned downstream from the second machine to the first machine

Figure 12G

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2015/020780

A. CLASSIFICATION OF SUBJECT MATTER
 INV. H04L29/08
 ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
 H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WEIXIONG RAO ET AL: "Optimal Resource Placement in Structured Peer-to-Peer Networks", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE SERVICE CENTER, LOS ALAMITOS, CA, US, vol. 21, no. 7, July 2010 (2010-07), pages 1011-1026, XP011295472, ISSN: 1045-9219	1-6, 8-11, 13-33, 36,37, 39,40, 42-44
Y A	page 1014, left-hand column, paragraph 3.1 - right-hand column, paragraph 3.2 page 1016, right-hand column, paragraph 3.3 page 1017, left-hand column, paragraph 3.4 - page 1018, left-hand column, paragraph 3.4.3; figure 4 ----- -/--	34,35 38,41



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

24 June 2015

Date of mailing of the international search report

02/07/2015

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
 NL - 2280 HV Rijswijk
 Tel. (+31-70) 340-2040,
 Fax: (+31-70) 340-3016

Authorized officer

Fournier, Christophe

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2015/020780

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2002/007404 A1 (VANGE MARK [CA] ET AL) 17 January 2002 (2002-01-17)	1,2,5,7, 12-15, 18,20-23
Y	abstract	34,35
A	paragraphs [0012], [0022] - [0024], [0035], [0044], [0072] -----	24,43
A	US 2010/306252 A1 (JARVIS STEVEN A [US] ET AL) 2 December 2010 (2010-12-02) abstract paragraphs [0051] - [0053]; claims 1-2 -----	1,3,11, 13,16, 20-24, 29,43
A	EP 2 493 118 A1 (NEC CORP [JP]) 29 August 2012 (2012-08-29) abstract; claim 1; figures 1,2,8 -----	24,43

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2015/020780

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2002007404	A1	17-01-2002	
		AU 5163601 A	30-10-2001
		AU 5164301 A	30-10-2001
		AU 5164401 A	30-10-2001
		AU 5353201 A	30-10-2001
		AU 5353301 A	30-10-2001
		AU 5353401 A	30-10-2001
		AU 5353601 A	30-10-2001
		AU 5353701 A	30-10-2001
		AU 5355901 A	30-10-2001
		AU 5361301 A	30-10-2001
		AU 5544101 A	30-10-2001
		AU 5705801 A	30-10-2001
		AU 5907401 A	30-10-2001
		AU 5907501 A	30-10-2001
		US 2002002602 A1	03-01-2002
		US 2002002603 A1	03-01-2002
		US 2002002611 A1	03-01-2002
		US 2002002618 A1	03-01-2002
		US 2002002622 A1	03-01-2002
		US 2002002625 A1	03-01-2002
		US 2002002636 A1	03-01-2002
		US 2002002686 A1	03-01-2002
		US 2002004796 A1	10-01-2002
		US 2002004816 A1	10-01-2002
		US 2002007404 A1	17-01-2002
		US 2002019853 A1	14-02-2002
		US 2002023159 A1	21-02-2002
		US 2002056006 A1	09-05-2002
		US 2002059170 A1	16-05-2002
		US 2006129697 A1	15-06-2006
		US 2010157998 A1	24-06-2010
		US 2011238860 A1	29-09-2011
		US 2011246665 A1	06-10-2011
		US 2011302321 A1	08-12-2011
		US 2012054308 A1	01-03-2012
		US 2012059907 A1	08-03-2012
		US 2014074981 A1	13-03-2014
		US 2015019753 A1	15-01-2015
		WO 0180002 A1	25-10-2001
		WO 0180003 A2	25-10-2001
		WO 0180004 A2	25-10-2001
		WO 0180014 A2	25-10-2001
		WO 0180024 A2	25-10-2001
		WO 0180033 A2	25-10-2001
		WO 0180062 A2	25-10-2001
		WO 0180063 A2	25-10-2001
		WO 0180064 A2	25-10-2001
		WO 0180093 A2	25-10-2001
		WO 0180515 A2	25-10-2001
		WO 0180516 A2	25-10-2001
		WO 0180517 A2	25-10-2001
		WO 0180524 A2	25-10-2001
US 2010306252	A1	02-12-2010	
		US 2010306252 A1	02-12-2010
		WO 2010136334 A1	02-12-2010
EP 2493118	A1	29-08-2012	
		CN 102693209 A	26-09-2012
		EP 2493118 A1	29-08-2012

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2015/020780

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		JP 5625998 B2	19-11-2014
		JP 2012174081 A	10-09-2012
		US 2012215876 A1	23-08-2012
