(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 October 2013 (17.10.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/154905 A1

(51) International Patent Classification:
H04W 24/04 (2009.01) *H04W 76/00* (2009.01)

(21) International Application Number:
PCT/US2013/035257

(22) International Filing Date:
4 April 2013 (04.04.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/621,667	9 April 2012 (09.04.2012)	US
13/856,366	3 April 2013 (03.04.2013)	US
13/856,691	4 April 2013 (04.04.2013)	US

(71) Applicant: **SEVEN NETWORKS, INC.** [US/US]; 2100 Seaport Boulevard, Suite 100, Redwood City, CA 94063 (US).

(72) Inventors: **LUNA, Michael**; 519 Curie Drive, San Jose, CA 95123 (US). **TERVAHAUTA, Mikko**; 2015 Castro St., San Francisco, CA 94131 (US). **PAN, Yixin**; 35679 Dante Place, Fremont, CA 94536 (US).

(74) Agents: **FU, Yenyun** et al.; Perkins Coie LLP, P.O. Box 1247, Seattle, WA 98111-1247 (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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, 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, 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, ML, MR, NE, SN, TD, TG).

Published:

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

(54) Title: A METHOD AND SYSTEM FOR MANAGEMENT OF A VIRTUAL NETWORK CONNECTION WITHOUT HEARTBEAT MESSAGES

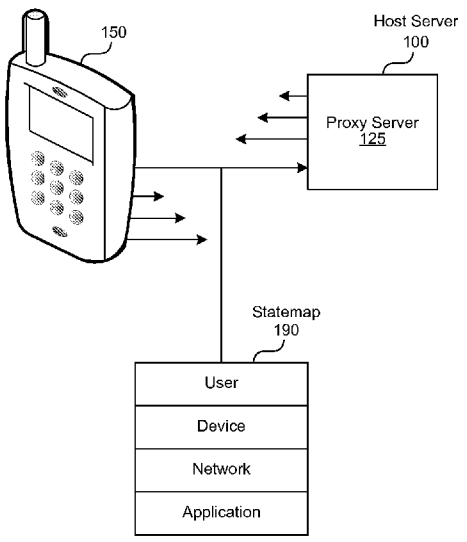


FIG. 1A

(57) Abstract: Systems and methods for management of a network connection without heartbeat messages are disclosed. One embodiment of a distributed proxy system performs a method for the communication of state between a client and a server in a distributed content delivery network using a state map. The state map sets a predicted communication correspondence frequency and thus eliminates the use or need of heart beat messages to manage session state and/or convey health status of system components.

WO 2013/154905 A1

A METHOD AND SYSTEM FOR MANAGEMENT OF A VIRTUAL NETWORK CONNECTION WITHOUT HEARTBEAT MESSAGES

CROSS-REFERENCE TO RELATED APPLICATIONS AND EFFECTIVE FILING DATE ENTITLEMENT

[0001] This application is entitled to the benefit of and/or the right of the priority to United States Provisional Application No. 61/621,667, entitled "METHOD AND SYSTEM FOR MANAGEMENT OF A VIRTUAL NETWORK CONNECTION WITHOUT HEARTBEAT," (Attorney Docket No. 76443-8126.US00), filed on April 9, 2012, and is hereby incorporated by reference in its entirety. This application is therefore entitled to an effective filing date of April 9, 2012.

[0002] This application is further entitled to the benefit of U.S. Patent Application No. 13/856,366 entitled "A METHOD AND SYSTEM FOR MANAGEMENT OF A VIRTUAL NETWORK CONNECTION WITHOUT HEARTBEAT MESSAGES," (Attorney Docket No. 76443-8126.US01), filed on April 3, 2013, which is entitled to the benefit of United States Provisional Application No. 61/621,667, entitled "METHOD AND SYSTEM FOR MANAGEMENT OF A VIRTUAL NETWORK CONNECTION WITHOUT HEARTBEAT," (Attorney Docket No. 76443-8126.US00), filed on April 9, 2012, and is hereby incorporated by reference in its entirety.

[0003] This application is further entitled to the benefit of and/or the right of the priority to U.S. Patent Application No. 13/856,691 entitled "MOBILE DEVICE AND METHOD FOR PRESERVING SESSION STATE INFORMATION WITHOUT A NEED FOR TRANSMISSION OF MESSAGES IN A WIRELESS NETWORK," (Attorney Docket No. 76443-8126.US02), filed on April 4, 2013, which is a continuation application and entitled to the benefit of U.S. Patent Application No. 13/856,366 entitled "A METHOD AND SYSTEM FOR MANAGEMENT OF A VIRTUAL NETWORK CONNECTION WITHOUT HEARTBEAT MESSAGES," (Attorney Docket No. 76443-8126.US01), filed on April 3, 2013, which is entitled to the benefit of United States Provisional Application No. 61/621,667, entitled "METHOD AND SYSTEM FOR MANAGEMENT OF A VIRTUAL NETWORK CONNECTION WITHOUT HEARTBEAT," (Attorney Docket No. 76443-8126.US00), filed on April 9, 2012, and is hereby incorporated by reference in its entirety.

BACKGROUND

[0004] To date, carriers have been able to invest in 4G and LTE networks to boost network capacity in hotspots. However, these solutions are reaching their limit. LTE and 4G are also showing that the perceived capacity of added bandwidth is causing users and applications to increase usage and data consumption. In the long run, it might add to the congestion problem rather than help.

[0005] While mobile or broadband networks may be designed for high-throughput of large amounts of data, they were not necessarily tailored to service the mobile applications that require frequent, low-throughput requests of small amounts of data. Existing networks also do not take into account different types of mobile traffic and priorities of the different types of traffic, for example, from a user experience perspective.

[0006] The issue has been exacerbated by the rapid increase of the popularity of applications with network-initiated functionalities, such as social media applications, cloud-based mobile clients, email clients, news feeds, status updates, multimedia content sharing, mobile games and other mobile applications, etc. Furthermore, the problem with constant polling is that mobile phones also rely on signaling to send and receive calls and SMS messages and basic mobile functions are forced to take a backseat to applications and other mobile clients.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] **FIG. 1A** illustrates an example diagram of a state map 190 used by a device 150 and proxy server 100 (or proxy server 125 of the host server 100) to convey and update system state to eliminate use of heart beat for session state and health manager.

[0008] **FIG. 1B** illustrates an example diagram of a system where a host server facilitates management of traffic, content caching, and/or resource conservation between mobile devices (e.g., wireless devices), an application server or content provider, or other servers such as an ad server, promotional content server, or an e-coupon server in a wireless network (or broadband network) for resource conservation. The host server can further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations. The host server can further manage a virtual network connection without heartbeat messages.

[0009] **FIG. 1C** illustrates an example diagram of a proxy and cache system distributed between the host server and device which facilitates network traffic management between a device, an application server or content provider, or other servers such as an ad server, promotional content server, or an e-coupon server for resource conservation and content caching. The proxy system distributed among the host server and the device can further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations. The proxy system distributed among the host server and the device can further manage a virtual network connection without heartbeat messages.

[0010] **FIG. 1D** illustrates an example diagram of the logical architecture of a distributed proxy and cache system.

[0011] **FIG. 1E** illustrates an example diagram showing the architecture of client side components in a distributed proxy and cache system.

[0012] **FIG. 1F** illustrates a diagram of the example components on the server side of the distributed proxy and cache system.

[0013] **FIG. 2A** depicts a block diagram illustrating another example of client-side components in a distributed proxy and cache system, further including a session state and/or health manager which is able to convey/manage state among mobile devices and the host server without using heart beat messages.

[0014] **FIG. 2B** depicts a block diagram illustrating additional components in the session state and/or health manager shown in the example of **FIG. 2A**.

[0015] **FIG. 3A** depicts a block diagram illustrating an example of server-side components in a distributed proxy and cache system, further including a session state and/or health manager which is able to convey/manage state among mobile devices and the host server without using heart beat messages.

[0016] **FIG. 3B** depicts a block diagram illustrating additional components in the session state and/or health manager shown in the example of **FIG. 3A**.

[0017] **FIG. 4A** depicts a block diagram illustrating an example of client-side components in a distributed proxy and cache system residing on a mobile device (e.g., wireless device) that

manages traffic in a wireless network (or broadband network) for resource conservation, content caching, and/or traffic management. The client-side proxy (or local proxy) can further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations.

[0018] **FIG. 4B** depicts a block diagram illustrating a further example of components in the cache system shown in the example of **FIG. 4A** which is capable of caching and adapting caching strategies for mobile application behavior and/or network conditions. Components capable of detecting long poll requests and managing caching of long polls are also illustrated.

[0019] **FIG. 4C** depicts a block diagram illustrating additional components in the application behavior detector and the caching policy manager in the cache system shown in the example of **FIG. 4A** which is further capable of detecting cache defeat and perform caching of content addressed by identifiers intended to defeat cache.

[0020] **FIG. 4D** depicts a block diagram illustrating examples of additional components in the local cache shown in the example of **FIG. 4A** which is further capable of performing mobile traffic categorization and policy implementation based on application behavior and/or user activity.

[0021] **FIG. 5A** depicts a block diagram illustrating an example of server-side components in a distributed proxy and cache system that manages traffic in a wireless network (or broadband network) for resource conservation, content caching, and/or traffic management. The server-side proxy (or proxy server) can further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations.

[0022] **FIG. 5B** depicts a block diagram illustrating a further example of components in the caching policy manager in the cache system shown in the example of **FIG. 5A** which is capable of caching and adapting caching strategies for mobile application behavior and/or network conditions. Components capable of detecting long poll requests and managing caching of long polls are also illustrated.

[0023] **FIG. 5C** depicts a block diagram illustrating another example of components in the proxy system shown in the example of **FIG. 5A** which is further capable of managing and detecting cache defeating mechanisms and monitoring content sources.

[0024] **FIG. 5D** depicts a block diagram illustrating examples of additional components in proxy server shown in the example of **FIG. 5A** which is further capable of performing mobile traffic categorization and policy implementation based on application behavior and/or traffic priority.

[0025] **FIG. 6A** depicts a flow diagram illustrating an example process for distributed content caching between a mobile device (e.g., any wireless device) and remote proxy and the distributed management of content caching.

[0026] **FIG. 6B** depicts a timing diagram showing how data requests from a mobile device (e.g., any wireless device) to an application server/content provider in a wireless network (or broadband network) can be coordinated by a distributed proxy system in a manner such that network and battery resources are conserved through using content caching and monitoring performed by the distributed proxy system.

[0027] **FIG. 7** depicts a table showing examples of different traffic or application category types which can be used in implementing network access and content delivery policies.

[0028] **FIG. 8** depicts a table showing examples of different content category types which can be used in implementing network access and content delivery policies.

[0029] **FIG. 9** depicts an interaction diagram showing how polls having data requests from a mobile device (e.g., any wireless device) to an application server/content provider over a wireless network (or broadband network) can be can be cached on the local proxy and managed by the distributed caching system.

[0030] **FIG. 10** depicts an interaction diagram showing how polls for content from an application server/content provider which employs cache-defeating mechanisms in identifiers (e.g., identifiers intended to defeat caching) over a wireless network (or broadband network) can be detected and locally cached.

[0031] **FIG. 11** depicts a flow chart illustrating an example process for collecting information about a request and the associated response to identify cacheability and caching the response.

[0032] **FIG. 12** depicts a flow chart illustrating an example process showing decision flows to determine whether a response to a request can be cached.

[0033] **FIG. 13** depicts a flow chart illustrating an example process for determining potential for cacheability based on request periodicity and/or response repeatability.

[0034] **FIG. 14** depicts a flow chart illustrating an example process for dynamically adjusting caching parameters for a given request or client.

[0035] **FIG. 15** depicts a flow chart illustrating example processes for application and/or traffic (data) categorization while factoring in user activity and expectations for implementation of network access and content delivery policies.

[0036] **FIG. 16A** depicts a flow chart illustrating example processes for handling traffic which is to be suppressed at least temporarily determined from application/traffic categorization.

[0037] **FIG. 16B** depicts a flow chart illustrating an example process for selection of a network configuration for use in sending traffic based on application and/or traffic (data) categorization.

[0038] **FIG. 16C** depicts a flow chart illustrating an example process for implementing network access and content delivery policies based on application and/or traffic (data) categorization.

[0039] **FIG. 17** depicts a flow chart illustrating an example process for network selection based on mobile user activity or user expectations.

[0040] **FIG. 18** depicts a data timing diagram showing an example of detection of periodic request which may be suitable for caching.

[0041] **FIG. 19** depicts a data timing diagram showing an example of detection of change in request intervals and updating of server polling rate in response thereto.

[0042] **FIG. 20** depicts a data timing diagram showing an example of serving foreground requests with cached entries.

[0043] **FIG. 21** depicts a data timing diagram showing an example of the possible effect of cache invalidation that occurs after outdated content has been served once again to a requesting application.

[0044] **FIG. 22** depicts a data timing diagram showing cache management and response taking into account the time-to-live (TTL) set for cache entries.

[0045] **FIG. 23** depicts a flow chart illustrating an example flow for generating state map for the device to determine session state at a device for determining whether activity occurred when expected or predicted.

[0046] **FIG. 24** depicts a flow chart illustrating an example flow for conserving resources of the network when maintaining or conveying the session state of the mobile device.

[0047] **FIG. 25** depicts a flow chart illustrating an example flow for quantitatively using a tolerable threshold of lack of communication to determine or assess the health status of a device based an amount of time form which a previous correspondence was received.

[0048] **FIG. 26** depicts a flow chart illustrating example processes for gathering state information used in assessing or determining device or session health state.

[0049] **FIG. 27** shows a diagrammatic representation of a machine in the example form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed.

DETAILED DESCRIPTION

[0050] The following description and drawings are illustrative and are not to be construed as limiting. Numerous specific details are described to provide a thorough understanding of the disclosure. However, in certain instances, well-known or conventional details are not described in order to avoid obscuring the description. References to "one embodiment" or "an embodiment" in the present disclosure can be, but not necessarily are, references to the same embodiment and such references mean at least one of the embodiments.

[0051] Reference in this specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the disclosure. The appearances of the

phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment, nor are separate or alternative embodiments mutually exclusive of other embodiments. Moreover, various features are described which may be exhibited by some embodiments and not by others. Similarly, various requirements are described which may be requirements for some embodiments but not other embodiments.

[0052] The terms used in this specification generally have their ordinary meanings in the art, within the context of the disclosure, and in the specific context where each term is used. Certain terms that are used to describe the disclosure are discussed below, or elsewhere in the specification, to provide additional guidance to the practitioner regarding the description of the disclosure. For convenience, certain terms may be highlighted, for example using italics and/or quotation marks. The use of highlighting has no influence on the scope and meaning of a term; the scope and meaning of a term is the same, in the same context, whether or not it is highlighted. It will be appreciated that same thing can be said in more than one way.

[0053] Consequently, alternative language and synonyms may be used for any one or more of the terms discussed herein, nor is any special significance to be placed upon whether or not a term is elaborated or discussed herein. Synonyms for certain terms are provided. A recital of one or more synonyms does not exclude the use of other synonyms. The use of examples anywhere in this specification, including examples of any terms discussed herein, is illustrative only, and is not intended to further limit the scope and meaning of the disclosure or of any exemplified term. Likewise, the disclosure is not limited to various embodiments given in this specification.

[0054] Without intent to limit the scope of the disclosure, examples of instruments, apparatus, methods and their related results according to the embodiments of the present disclosure are given below. Note that titles or subtitles may be used in the examples for convenience of a reader, which in no way should limit the scope of the disclosure. Unless otherwise defined, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this disclosure pertains. In the case of conflict, the present document, including definitions, will control.

[0055] Embodiments of the present disclosure include systems and methods for preserving session state information among devices via a wireless network without using heat beat messages or other messages.

[0056] There are multiple factors that contribute to the proliferation of data: the end-user, mobile devices, wireless devices, mobile applications, and the network. As mobile devices evolve, so do the various elements associated with them, including availability, applications, user behavior, and location, thus changing the way the network interacts with the device and the application.

[0057] The disclosed technology provides a comprehensive and end-to-end solution that is able to address each element for operators and devices manufacturers to support both the shift in mobile or wireless devices and the surge in data by leveraging the premise that mobile content has a definable or relevant "freshness" value. The "freshness" of mobile content can be determined, either with certainty, or with some heuristics having a tolerance within which the user experience is enhanced, or not negatively impacted, or negatively impacted but is either not perceptible to the user or within a tolerable threshold level.

[0058] The disclosed innovation transparently determines such "freshness" by monitoring, analyzing, and applying rules (which may be heuristically determined) to the transactions (requests/responses) between applications (e.g., mobile applications) and the peers (corresponding server or other clients). Moreover, the technology is further able to effectively cache content which may be marked by its originating/host server as being "non-cacheable" and identify some "freshness" value which can then be used in implementing application-specific caching. In general, the "freshness" value has an approximate minimum value which is typically determined using the update interval (e.g., interval with which requests are sent) between the application and its corresponding server/host.

[0059] One embodiment of the disclosed technology includes a system that optimizes multiple aspects of the connection with wired and wireless networks and devices through a comprehensive view of device and application activity including: loading, current application needs on a device, controlling the type of access (push vs. pull or hybrid), location, concentration of users in a single area, time of day, how often the user interacts with the application, content or device, and using this information to shape traffic to a cooperative client/server or simultaneously mobile devices without a cooperative client. Because the disclosed server is not tied to any specific network provider, it has visibility into the network performance across all service providers. This enables optimizations to be applied to devices regardless of the operator or service provider, thereby enhancing the user experience and managing network utilization while roaming. Bandwidth has been considered a major issue in

wireless networks today. More and more research has been done related to the need for additional bandwidth to solve access problems. Many of the performance enhancing solutions and next generation standards, such as those commonly referred to as 3.5G, LTE, 4G, and WiMAX, are focused on providing increased bandwidth. Although partially addressed by the standards, a key problem that remains is lack of bandwidth on the signaling channel more so than the data channel and the standard does not address battery life very well.

[0060] Embodiments of the disclosed technology include, for example, alignment of requests from multiple applications to minimize the need for several polling requests; leverage specific content types to determine how to proxy/manage a connection/content; and applying specific heuristics associated with device, user behavioral patterns (how often they interact with the device/application) and/or network parameters.

[0061] Embodiments of the present technology can further include moving recurring HTTP polls performed by various widgets, RSS readers, etc., to remote network node (e.g., Network Operation Center (NOC)), thus considerably lowering device battery/power consumption, radio channel signaling and bandwidth usage. Additionally, the offloading can be performed transparently so that existing applications do not need to be changed.

[0062] In some embodiments, this can be implemented using a local proxy on the mobile device (e.g., any wireless device) which automatically detects recurring requests for the same content (RSS feed, Widget data set) that matches a specific rule (e.g., happens every 15 minutes). The local proxy can automatically cache the content on the mobile device while delegating the polling to the server (e.g., a proxy server operated as an element of a communications network). The server can then notify the mobile/client proxy if the content changes, and if content has not changed (or not changed sufficiently, or in an identified manner or amount) the mobile proxy provides the latest version in its cache to the user (without need to utilize the radio at all). This way the mobile or wireless device (e.g., a mobile phone, smart phone, M2M module/MODEM, or any other wireless devices, etc.) does not need to open (e.g., thus powering on the radio) or use a data connection if the request is for content that is monitored and that has been not flagged as new/changed.

[0063] The logic for automatically adding content sources/application servers (e.g., including URLs/content) to be monitored can also check for various factors like how often the content is the same, how often the same request is made (is there a fixed interval/pattern?),

which application is requesting the data, etc. Similar rules to decide between using the cache and request the data from the original source may also be implemented and executed by the local proxy and/or server.

[0064] For example, when the request comes at an unscheduled/unexpected time (user initiated check), or after every (n) consecutive times the response has been provided from the cache, etc., or if the application is running in the background vs. in a more interactive mode of the foreground. As more and more mobile applications or wireless enabled applications base their features on resources available in the network, this becomes increasingly important. In addition, the disclosed technology allows elimination of unnecessary chatter from the network, benefiting the operators trying to optimize the wireless spectrum usage.

Traffic Categorization and Policy

[0065] In some embodiments, the disclosed proxy system is able to establish policies for choosing traffic (data, content, messages, updates, etc.) to cache and/or shape. Additionally, by combining information from observing the application making the network requests, getting explicit information from the application, or knowing the network destination the application is reaching, the disclosed technology can determine or infer what category the transmitted traffic belongs to.

[0066] For example, in one embodiment, mobile or wireless traffic can be categorized as: (a1) interactive traffic or (a2) background traffic. The difference is that in (a1) a user is actively waiting for a response, while in (a2) a user is not expecting a response. This categorization can be used in conjunction with or in lieu of a second type of categorization of traffic: (b1) immediate, (b2) low priority, or (b3) immediate if the requesting application is in the foreground and active.

[0067] For example, a new update, message or email may be in the (b1) category to be delivered immediately, but it still is (a2) background traffic — a user is not actively waiting for it. A similar categorization applies to instant messages when they come outside of an active chat session. During an active chat session a user is expecting a response faster. Such user expectations are determined or inferred and factored into when optimizing network use and device resources in performing traffic categorization and policy implementation.

[0068] Some examples of the applications of the described categorization scheme, include the following: (a1) interactive traffic can be categorized as (b1) immediate — but (a2) background traffic may also be (b2) or (b3). An example of a low priority transfer is email or message maintenance transaction such as deleting email or other messages or marking email as read at the mail or application server. Such a transfer can typically occur at the earlier of (a) timer exceeding a timeout value (for example, 2 minutes), and (b) data being sent for other purposes.

[0069] An example of (b3) is IM presence updates, stock ticker updates, weather updates, status updates, news feeds. When the UI of the application is in the foreground and/or active (for example, as indicated by the backlight of the device/phone being lit or as determined or inferred from the status of other sensors), updates can be considered immediate whenever server has something to push to the device. When the application is not in the foreground or not active, such updates can be suppressed until the application comes to foreground and is active.

[0070] With some embodiments, networks can be selected or optimized simultaneously for (a1) interactive traffic and (a2) background traffic.

[0071] In some embodiments, as the wireless device or mobile device proxy (separately or in conjunction with the server proxy) is able to categorize the traffic as (for example) (a1) interactive traffic or (a2) background traffic, it can apply different policies to different types of traffic. This means that it can internally operate differently for (a1) and (a2) traffic (for example, by allowing interactive traffic to go through to the network in whole or in part, and apply stricter traffic control to background traffic; or the device side only allows a request to activate the radio if it has received information from the server that the content at the host has been updated, etc.).

[0072] When the request does require access over the wireless network, the disclosed technology can request the radio layer to apply different network configurations to different traffic. Depending on the type of traffic and network this may be achieved by different means:

[0073] (1) Using 3G/4G for (a1) and 2G/2.5G for (a2);

[0074] (2) Explicitly specifying network configuration for different data sets (e.g., in terms of use of FACH (forward access channel) vs. DCH (dedicated channel), or otherwise requesting lower/more network efficient data rates for background traffic); or

[0075] (3) Utilizing different network access points for different data sets (access points which would be configured to use network resources differently similar to (1) and (2) above).

[0076] Additionally, 3GPP Fast Dormancy calls for improvements so that applications, operating systems or the mobile device would have awareness of the traffic type to be more efficient in the future. Embodiments of the disclosed system, having the knowledge of the traffic category and being able to utilize Fast Dormancy appropriately, may solve the problem identified in Fast Dormancy. This way the mobile or broadband network does not need to be configured with a compromised configuration that adversely impacts both battery consumption and network signaling resources.

Polling schedule

[0077] Detecting (or determining) a polling schedule allows the proxy server (server-side of the distributed cache system) to be as close as possible with its polls to the application polls. Many applications employ scheduled interval polling (e.g., every 4 hours or every 30 seconds, at another time interval). The client side proxy can detect automatic polls based on time measurements and create a automatic polling profile for an application. As an example, the local proxy attempts to detect the time interval between requests and after 2, 3, 4, or more polls, and determines an automatic rate if the time intervals are all within 1 second (or another measure of relative closeness) of each other. If not, the client may collect data from a greater number of polling events (e.g., 10-12 polls) and apply a statistical analysis to determine, compute, or estimate a value for the average interval that is used. The polling profile is delivered to the server where it is used. If it is a frequent manual request, the locally proxy can substitute it with a default interval for this application taken from a profile for non-critical applications.

[0078] In some embodiments, the local proxy (e.g., device side proxy) may keep monitoring the application/client polls and update the polling interval. If it changes by more than 30% (or another predetermined/dynamic/conditional value) from the current value, it is communicated to the proxy server (e.g., server-side proxy). This approach can be referred to

as the scenario of "lost interest." In some instances, the local proxy can recognize requests made outside of this schedule, consider them "manual," and treat them accordingly.

Application classes/Modes of caching

[0079] In some embodiments, applications can be organized into three groups or modes of caching. Each mobile client/application can be categorized to be treated as one of these modes, or treated using multiple modes, depending on one or more conditions.

[0080] A) Fully cached — local proxy updates (e.g., sends application requests directly over the network to be serviced by the application server/content host) only when the proxy server tells the local proxy to update. In this mode, the local proxy can ignore manual requests and the proxy server uses the detected automatic profile (e.g., sports score applets, Facebook, every 10, 15, 30, or more polls) to poll the application server/content provider.

[0081] B) Partially cached — the local proxy uses the local or internal cache for automatic requests (e.g., application automatic refreshes), other scheduled requests but passes through some manual requests (e.g., email download, Ebay or some Facebook requests); and

[0082] C) Never cached (e.g., real-time stock ticker, sports scores/statuses; however, in some instances, 15 minutes delayed quotes can be safely placed on 30 seconds schedules — B or even A).

[0083] The actual application or caching mode classification can be determined based on the rate of content change and critical character of data. Unclassified applications by default can be set as class C.

Backlight and active applications

[0084] In some embodiments, the local proxy starts by detecting the device backlight status. Requests made with the screen light 'off' can be allowed to use the local cache if a request with identical signature is registered with the proxy server, which is polling the original host server/content server(s) to which the requests are directed. If the screen light is 'on,' further detection can be made to determine whether it is a background application or for other indicators that local cache entries can or cannot be used to satisfy the request. When identified, the requests for which local entries can be used may be processed identically to the screen light

off situation. Foreground requests can use the aforementioned application classification to assess when cached data is safe to use to process requests.

[0085] **FIG. 1A-1** illustrates an example diagram of a state map 190 used by a device 150 and proxy server 100 (or proxy server 125 of the host server 100) to convey and update system state to eliminate use of heart beat for session state and health manager.

[0086] The state map can be constructed or generated by the mobile device 150 (e.g., device side local proxy 175) based on system state information aggregated using knowledge of either the device 150 and/or the server (proxy server 175) on server 100. The system state information in the state map 190 is typically a function of any or all of, user information, device 150 information, network information, and/or application information. The system state information that is aggregated can generally include any combination of or all information aggregated by the components shown in the examples of **FIG. 2-FIG. 5**.

[0087] The state map 190 conveys state information between the device 150 (e.g., via the local proxy 175) and the proxy server 125 of server 100, including but not limited to user behavior, user pattern, application type, application pattern, network capacity, network outage, or any other information which affects how frequently and when data is requested at the mobile device, either in the foreground or in the background. Based on this state map 190, the system (incl. proxy server 125 and mobile device 150) has an awareness of when activity is expected between the proxy and the device and when no activity is expected. Based on this awareness, the system now needs not use keep alive messages to acknowledge or convey health status. This is beneficial for further conserving system and network resources in that additional messages need not periodically be sent for conveying information about session state and/or conveying information about health of either the local proxy 175 of the mobile device 150 and/or the proxy server 125 of the server 100.

[0088] In using the state map 190, the local proxy 175 can determine when or predict when correspondence or data should be received. If data or communication is not received within some predicted time frame, one party can request the information, or request to reset the system/cache. The predicted time frame can be represented, for example, by a tolerable threshold of no communication which may have one or more values. The values may differ based on time of day, or what applications are installed, what applications are running, day of week, the user's behavior/habits, etc.

[0089] **FIG. 1B** illustrates an example diagram of a system where a host server 100 facilitates management of traffic, content caching, and/or resource conservation between mobile devices (e.g., wireless devices 150), and an application server or content provider 110, or other servers such as an ad server 120A, promotional content server 120B, or an e-coupon server 120C in a wireless network (or broadband network) for resource conservation. The host server can further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations.

[0090] The client devices 150 can be any system and/or device, and/or any combination of devices/systems that is able to establish a connection, including wired, wireless, cellular connections with another device, a server and/or other systems such as host server 100 and/or application server/content provider 110. Client devices 150 will typically include a display and/or other output functionalities to present information and data exchanged between among the devices 150 and/or the host server 100 and/or application server/content provider 110. The application server/content provider 110 can by any server including third party servers or service/content providers further including advertisement, promotional content, publication, or electronic coupon servers or services. Similarly, separate advertisement servers 120A, promotional content servers 120B, and/or e-Coupon servers 120C as application servers or content providers are illustrated by way of example.

[0091] For example, the client devices 150 can include mobile, hand held or portable devices, wireless devices, or non-portable devices and can be any of, but are not limited to, a server desktop, a desktop computer, a computer cluster, or portable devices, including a notebook, a laptop computer, a handheld computer, a palmtop computer, a mobile phone, a cell phone, a smart phone, a PDA, a Blackberry device, a Palm device, a handheld tablet (e.g., an iPad or any other tablet), a hand held console, a hand held gaming device or console, any SuperPhone such as the iPhone, and/or any other portable, mobile, hand held devices, or fixed wireless interface such as a M2M device, etc. In one embodiment, the client devices 150, host server 100, and application server 110 are coupled via a network 106 and/or a network 108. In some embodiments, the devices 150 and host server 100 may be directly connected to one another.

[0092] The input mechanism on client devices 150 can include a touch screen keypad (including single touch, multi-touch, gesture sensing in 2D or 3D, etc.), a physical keypad, a mouse, a pointer, a track pad, motion detector (e.g., including 1-axis, 2-axis, 3-axis

accelerometer, etc.), a light sensor, capacitance sensor, resistance sensor, temperature sensor, proximity sensor, a piezoelectric device, device orientation detector (e.g., electronic compass, tilt sensor, rotation sensor, gyroscope, accelerometer), or a combination of the above.

[0093] Signals received or detected indicating user activity at client devices 150 through one or more of the above input mechanism, or others, can be used in the disclosed technology in acquiring context awareness at the client device 150. Context awareness at client devices 150 generally includes, by way of example but not limitation, client device 150 operation or state acknowledgement, management, user activity/behavior/interaction awareness, detection, sensing, tracking, trending, and/or application (e.g., mobile applications) type, behavior, activity, operating state, etc.

[0094] Context awareness in the present disclosure also includes knowledge and detection of network side contextual data and can include network information such as network capacity, bandwidth, traffic, type of network/connectivity, and/or any other operational state data. Network side contextual data can be received from and/or queried from network service providers (e.g., cell provider 112 and/or Internet service providers) of the network 106 and/or network 108 (e.g., by the host server and/or devices 150). In addition to application context awareness as determined from the client 150 side, the application context awareness may also be received from or obtained/queried from the respective application/service providers 110 (by the host 100 and/or client devices 150).

[0095] The host server 100 can use, for example, contextual information obtained for client devices 150, networks 106/108, applications (e.g., mobile applications), application server/provider 110, or any combination of the above, to manage the traffic in the system to satisfy data needs of the client devices 150 (e.g., to satisfy application or any other request including HTTP request). In one embodiment, the traffic is managed by the host server 100 to satisfy data requests made in response to explicit or non-explicit user 103 requests and/or device/application maintenance tasks. The traffic can be managed such that network consumption, for example, use of the cellular network is conserved for effective and efficient bandwidth utilization. In addition, the host server 100 can manage and coordinate such traffic in the system such that use of device 150 side resources (e.g., including but not limited to battery power consumption, radio use, processor/memory use) are optimized with a general philosophy for resource conservation while still optimizing performance and user experience.

[0096] For example, in context of battery conservation, the device 150 can observe user activity (for example, by observing user keystrokes, backlight status, or other signals via one or more input mechanisms, etc.) and alters device 150 behaviors. The device 150 can also request the host server 100 to alter the behavior for network resource consumption based on user activity or behavior.

[0097] In one embodiment, the traffic management for resource conservation is performed using a distributed system between the host server 100 and client device 150. The distributed system can include proxy server and cache components on the server side 100 and on the device/client side, for example, as shown by the server cache 135 on the server 100 side and the local cache 185 on the client 150 side.

[0098] Functions and techniques disclosed for context aware traffic management for resource conservation in networks (e.g., network 106 and/or 108) and devices 150, reside in a distributed proxy and cache system. The proxy and cache system can be distributed between, and reside on, a given client device 150 in part or in whole and/or host server 100 in part or in whole. The distributed proxy and cache system are illustrated with further reference to the example diagram shown in **FIG. 1B**. Functions and techniques performed by the proxy and cache components in the client device 150, the host server 100, and the related components therein are described, respectively, in detail with further reference to the examples of **FIG. 2-3**.

[0099] In one embodiment, client devices 150 communicate with the host server 100 and/or the application server 110 over network 106, which can be a cellular network and/or a broadband network. To facilitate overall traffic management between devices 150 and various application servers/content providers 110 to implement network (bandwidth utilization) and device resource (e.g., battery consumption), the host server 100 can communicate with the application server/providers 110 over the network 108, which can include the Internet (e.g., a broadband network).

[00100] In general, the networks 106 and/or 108, over which the client devices 150, the host server 100, and/or application server 110 communicate, may be a cellular network, a broadband network, a telephonic network, an open network, such as the Internet, or a private network, such as an intranet and/or the extranet, or any combination thereof. For example, the Internet can provide file transfer, remote log in, email, news, RSS, cloud-based services, instant messaging, visual voicemail, push mail, VoIP, and other services through any known or

convenient protocol, such as, but is not limited to the TCP/IP protocol, UDP, HTTP, DNS, FTP, UPnP, NSF, ISDN, PDH, RS-232, SDH, SONET, etc.

[00101] The networks 106 and/or 108 can be any collection of distinct networks operating wholly or partially in conjunction to provide connectivity to the client devices 150 and the host server 100 and may appear as one or more networks to the serviced systems and devices. In one embodiment, communications to and from the client devices 150 can be achieved by an open network, such as the Internet, or a private network, broadband network, such as an intranet and/or the extranet. In one embodiment, communications can be achieved by a secure communications protocol, such as secure sockets layer (SSL), or transport layer security (TLS).

[00102] In addition, communications can be achieved via one or more networks, such as, but are not limited to, one or more of WiMax, a Local Area Network (LAN), Wireless Local Area Network (WLAN), a Personal area network (PAN), a Campus area network (CAN), a Metropolitan area network (MAN), a Wide area network (WAN), a Wireless wide area network (WWAN), or any broadband network, and further enabled with technologies such as, by way of example, Global System for Mobile Communications (GSM), Personal Communications Service (PCS), Bluetooth, Wi-Fi, Fixed Wireless Data, 2G, 2.5G, 3G, 4G, IMT-Advanced, pre-4G, LTE Advanced, mobile WiMax, WiMax 2, WirelessMAN-Advanced networks, enhanced data rates for GSM evolution (EDGE), General packet radio service (GPRS), enhanced GPRS, iBurst, UMTS, HSPDA, HSUPA, HSPA, UMTS-TDD, 1xRTT, EV-DO, messaging protocols such as, TCP/IP, SMS, MMS, extensible messaging and presence protocol (XMPP), real time messaging protocol (RTMP), instant messaging and presence protocol (IMPP), instant messaging, USSD, IRC, or any other wireless data networks, broadband networks, or messaging protocols.

[00103] **FIG. 1C** illustrates an example diagram of a proxy and cache system distributed between the host server 100 and device 150 which facilitates network traffic management between the device 150 and an application server or content provider 110, or other servers such as an ad server 120A, promotional content server 120B, or an e-coupon server 120C for resource conservation and content caching. The proxy system distributed among the host server 100 and the device 150 can further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations.

[00104] The distributed proxy and cache system can include, for example, the proxy server 125 (e.g., remote proxy) and the server cache 135 components on the server side. The server-side proxy 125 and cache 135 can, as illustrated, reside internal to the host server 100. In addition, the proxy server 125 and cache 135 on the server-side can be partially or wholly external to the host server 100 and in communication via one or more of the networks 106 and 108. For example, the proxy server 125 may be external to the host server and the server cache 135 may be maintained at the host server 100. Alternatively, the proxy server 125 may be within the host server 100 while the server cache is external to the host server 100. In addition, each of the proxy server 125 and the cache 135 may be partially internal to the host server 100 and partially external to the host server 100. The application server/content provider 110 can by any server including third party servers or service/content providers further including advertisement, promotional content, publication, or electronic coupon servers or services. Similarly, separate advertisement servers 120A, promotional content servers 120B, and/or e-Coupon servers 120C as application servers or content providers are illustrated by way of example.

[00105] The distributed system can also, include, in one embodiment, client-side components, including by way of example but not limitation, a local proxy 175 (e.g., a mobile client on a mobile device) and/or a local cache 185, which can, as illustrated, reside internal to the device 150 (e.g., a mobile device).

[00106] In addition, the client-side proxy 175 and local cache 185 can be partially or wholly external to the device 150 and in communication via one or more of the networks 106 and 108. For example, the local proxy 175 may be external to the device 150 and the local cache 185 may be maintained at the device 150. Alternatively, the local proxy 175 may be within the device 150 while the local cache 185 is external to the device 150. In addition, each of the proxy 175 and the cache 185 may be partially internal to the host server 100 and partially external to the host server 100.

[00107] In one embodiment, the distributed system can include an optional caching proxy server 199. The caching proxy server 199 can be a component which is operated by the application server/content provider 110, the host server 100, or a network service provider 112, and or any combination of the above to facilitate network traffic management for network and device resource conservation. Proxy server 199 can be used, for example, for caching content to be provided to the device 150, for example, from one or more of, the application

server/provider 110, host server 100, and/or a network service provider 112. Content caching can also be entirely or partially performed by the remote proxy 125 to satisfy application requests or other data requests at the device 150.

[00108] In context aware traffic management and optimization for resource conservation in a network (e.g., cellular or other wireless networks), characteristics of user activity/behavior and/or application behavior at a mobile device (e.g., any wireless device) 150 can be tracked by the local proxy 175 and communicated, over the network 106 to the proxy server 125 component in the host server 100, for example, as connection metadata. The proxy server 125 which in turn is coupled to the application server/provider 110 provides content and data to satisfy requests made at the device 150.

[00109] In addition, the local proxy 175 can identify and retrieve mobile device properties, including one or more of, battery level, network that the device is registered on, radio state, or whether the mobile device is being used (e.g., interacted with by a user). In some instances, the local proxy 175 can delay, expedite (prefetch), and/or modify data prior to transmission to the proxy server 125, when appropriate, as will be further detailed with references to the description associated with the examples of **FIG. 2-5**.

[00110] The local database 185 can be included in the local proxy 175 or coupled to the local proxy 175 and can be queried for a locally stored response to the data request prior to the data request being forwarded on to the proxy server 125. Locally cached responses can be used by the local proxy 175 to satisfy certain application requests of the mobile device 150, by retrieving cached content stored in the cache storage 185, when the cached content is still valid.

[00111] Similarly, the proxy server 125 of the host server 100 can also delay, expedite, or modify data from the local proxy prior to transmission to the content sources (e.g., the application server/content provider 110). In addition, the proxy server 125 uses device properties and connection metadata to generate rules for satisfying request of applications on the mobile device 150. The proxy server 125 can gather real time traffic information about requests of applications for later use in optimizing similar connections with the mobile device 150 or other mobile devices.

[00112] In general, the local proxy 175 and the proxy server 125 are transparent to the multiple applications executing on the mobile device. The local proxy 175 is generally

transparent to the operating system or platform of the mobile device and may or may not be specific to device manufacturers. In some instances, the local proxy 175 is optionally customizable in part or in whole to be device specific. In some embodiments, the local proxy 175 may be bundled into a wireless model, a firewall, and/or a router.

[00113] In one embodiment, the host server 100 can in some instances, utilize the store and forward functions of a short message service center (SMSC) 112, such as that provided by the network service provider, in communicating with the device 150 in achieving network traffic management. Note that SMSC 112 can also utilize any other type of alternative channel including USSD or other network control mechanisms. As will be further described with reference to the example of **FIG. 3**, the host server 100 can forward content or HTTP responses to the SMSC 112 such that it is automatically forwarded to the device 150 if available, and for subsequent forwarding if the device 150 is not currently available.

[00114] In general, the disclosed distributed proxy and cache system allows optimization of network usage, for example, by serving requests from the local cache 185, the local proxy 175 reduces the number of requests that need to be satisfied over the network 106. Further, the local proxy 175 and the proxy server 125 may filter irrelevant data from the communicated data. In addition, the local proxy 175 and the proxy server 125 can also accumulate low priority data and send it in batches to avoid the protocol overhead of sending individual data fragments. The local proxy 175 and the proxy server 125 can also compress or transcode the traffic, reducing the amount of data sent over the network 106 and/or 108. The signaling traffic in the network 106 and/or 108 can be reduced, as the networks are now used less often and the network traffic can be synchronized among individual applications.

[00115] With respect to the battery life of the mobile device 150, by serving application or content requests from the local cache 185, the local proxy 175 can reduce the number of times the radio module is powered up. The local proxy 175 and the proxy server 125 can work in conjunction to accumulate low priority data and send it in batches to reduce the number of times and/or amount of time when the radio is powered up. The local proxy 175 can synchronize the network use by performing the batched data transfer for all connections simultaneously.

[00116] **FIG. 1D** illustrates an example diagram of the logical architecture of a distributed proxy and cache system.

[00117] The distributed system can include, for example the following components:

[00118] Client Side Proxy 175: a component installed in the Smartphone, mobile device or wireless device 150 that interfaces with device's operating system, as well as with data services and applications installed in the device. The client side proxy 175 is typically compliant with and able to operate with standard or state of the art networking protocols. Additional components and features of the client-side proxy 175 are illustrated with further references to the examples of **FIG. 2A-FIG. 2B** and **FIG. 4A-4D**.

[00119] The server side proxy 125 can include one or more servers that can interface with third party application servers (e.g., 199), mobile operator's network (which can be proxy 199 or an additional server that is not illustrated) and/or the client side proxy 175. In general, the server side proxy 125 can be compliant with and is generally able to operate with standard or state of the art networking protocols and/or specifications for interacting with mobile network elements and/or third party servers. Additional components and features of the server-side proxy 125 are illustrated with further references to the examples of **FIG. 3A-FIG. 3B** and **FIG. 5A-5D**.

[00120] Reporting and Usage Analytics Server 174: The Reporting and Usage Analytics system or component 174 can collect information from the client side 175 and/or the server side 125 and provides the necessary tools for producing reports and usage analytics can used for analyzing traffic and signaling data. Such analytics can be used by the proxy system in managing/reducing network traffic or by the network operator in monitoring their networks for possible improvements and enhancements. Note that the reporting and usage analytics system/component 174 as illustrated, may be a server separate from the server-side proxy 125, or it may be a component of the server-side proxy 125, residing partially or wholly therein.

[00121] **FIG. 1E** illustrates an example diagram showing the architecture of client side components in a distributed proxy and cache system.

[00122] The client side components 175 can include software components or agents installed on the mobile device that enables traffic optimization and performs the related functionalities on the client side. Components of the client side proxy 175 can operate transparently for end users and applications 163. The client side proxy 175 can be installed on mobile devices for optimization to take place, and it can effectuate changes on the data

routes. Once data routing is modified, the client side proxy 175 can respond to application requests to service providers or host servers, in addition to or instead of letting those applications 163 access data network directly. In general, applications 163 on the mobile device will not notice that the client side proxy 175 is responding to their requests. Some example components of the client side proxy 175 are described as follows:

[00123] Device State Monitor 121: The device state monitor 121 can be responsible for identifying several states and metrics in the device, such as network status, display status, battery level, etc. such that the remaining components in the client side proxy 175 can operate and make decisions according to device state, acting in an optimal way in each state.

[00124] Traffic Recognizer 122: The traffic recognizer 122 analyzes all traffic between the wireless device applications 163 and their respective host servers in order to identify recurrent patterns. Supported transport protocols include, for example, DNS, HTTP and HTTPS, such that traffic through those ports is directed to the client side proxy 175. While analyzing traffic, the client side proxy 175 can identify recurring polling patterns which can be candidates to be performed remotely by the server side proxy 125, and send to the protocol optimizer 123.

[00125] Protocol Optimizer 123: The protocol optimizer 123 can implement the logic of serving recurrent request from the local cache 185 instead of allowing those request go over the network to the service provider/application host server. One is its tasks is to eliminate or minimize the need to send requests to the network, positively affecting network congestion and device battery life.

[00126] Local Cache 185: The local cache 185 can store responses to recurrent requests, and can be used by the Protocol Optimizer 123 to send responses to the applications 163.

[00127] Traffic Scheduler 124: The traffic scheduler 124 can temporally move communications to optimize usage of device resources by unifying keep-alive signaling so that some or all of the different applications 163 can send keep-alive messages at the same time (traffic pipelining). Traffic scheduler124 may also decide to delay transmission of data that is not relevant at a given time (for example, when the device is not actively used).

[00128] Policy Manager 125: The policy manager 125 can store and enforce traffic optimization and reporting policies provisioned by a Policy Management Server (PMS). At the client side proxy 175 first start, traffic optimization and reporting policies (policy profiles) that is to be enforced in a particular device can be provisioned by the Policy Management Server.

[00129] Watch Dog 127: The watch dog 127 can monitor the client side proxy 175 operating availability. In case the client side proxy 175 is not working due to a failure or because it has been disabled, the watchdog 127 can reset DNS routing rules information and can restore original DNS settings for the device to continue working until the client side proxy 175 service is restored.

[00130] Reporting Agent 126: The reporting agent 126 can gather information about the events taking place in the device and sends the information to the Reporting Server. Event details are stored temporarily in the device and transferred to reporting server only when the data channel state is active. If the client side proxy 175 doesn't send records within twenty-four hours, the reporting agent 126 may attempt to open the connection and send recorded entries or, in case there are no entries in storage, an empty reporting packet. All reporting settings are configured in the policy management server.

[00131] Push Client 128: The push client 128 can be responsible for the traffic to between the server side proxy 125 and the client side proxy 175. The push client 128 can send out service requests like content update requests and policy update requests, and receives updates to those requests from the server side proxy 125. In addition, push client 128 can send data to a reporting server (e.g., the reporting and/or usage analytics system which may be internal to or external to the server side proxy 125).

[00132] The proxy server 199 has a wide variety of uses, from speeding up a web server by caching repeated requests, to caching web, DNS and other network lookups for a group of clients sharing network resources. The proxy server 199 is optional. The distributed proxy and cache system (125 and/or 175) allows for a flexible proxy configuration using either the proxy 199, additional proxy(s) in operator's network, or integrating both proxies 199 and an operator's or other third-party's proxy.

[00133] **FIG. 1F** illustrates a diagram of the example components on the server side of the distributed proxy and cache system.

[00134] The server side 125 of the distributed system can include, for example a relay server 142, which interacts with a traffic harmonizer 144, a polling server 145 and/or a policy management server 143. Each of the various components can communicate with the client side proxy 175, or other third party (e.g., application server/service provider 110 and/or other proxy 199) and/or a reporting and usage analytics system. Some example components of the server side proxy 125 is described as follows:

[00135] Relay Server 142: The relay server 142 is the routing agent in the distributed proxy architecture. The relay server 142 manages connections and communications with components on the client-side proxy 175 installed on devices and provides an administrative interface for reports, provisioning, platform setup, and so on.

[00136] Notification Server 141: The notification server 141 is a module able to connect to an operator's SMSC gateways and deliver SMS notifications to the client-side proxy 175. SMS notifications can be used when an IP link is not currently active, in order to avoid the client-side proxy 175 from activating a connection over the wireless data channel, thus avoiding additional signaling traffic. However, if the IP connection happens to be open for some other traffic, the notification server 141 can use it for sending the notifications to the client-side proxy 175. The user database can store operational data including endpoint (MSISDN), organization and Notification server 141 gateway for each resource (URIs or URLs).

[00137] Traffic Harmonizer 144: The traffic harmonizer 144 can be responsible for communication between the client-side proxy 175 and the polling server 145. The traffic harmonizer 144 connects to the polling server 145 directly or through the data storage 130, and to the client over any open or proprietary protocol such as the 7TP, implemented for traffic optimization. The traffic harmonizer 144 can be also responsible for traffic pipelining on the server side: if there's cached content in the database for the same client, this can be sent over to the client in one message.

[00138] Polling Server 145: The polling server 145 can poll third party application servers on behalf of applications that are being optimized). If a change occurs (i.e. new data available) for an application, the polling server 145 can report to the traffic harmonizer 144 which in turn sends a notification message to the client-side proxy 175 for it to clear the cache and allow application to poll application server directly.

[00139] Policy Management Server 143: The policy management server (PMS) 143 allows administrators to configure and store policies for the client-side proxies 175 (device clients). It also allows administrators to notify the client-side proxies 175 about policy changes. Using the policy management server 143, each operator can configure the policies to work in the most efficient way for the unique characteristics of each particular mobile operator's network.

[00140] Reporting and Usage Analytics Component: The Reporting and Usage Analytics component or system collects information from the client side 175 and/or from the server side 125, and provides the tools for producing reports and usage analytics that operators can use for analyzing application signaling and data consumption.

[00141] **FIG. 2A** depicts a block diagram illustrating another example of client-side components in a distributed proxy and cache system, further including a session state and/or health manager 401 which is able to convey/manage state among mobile devices and the host server without using heart beat messages. **FIG. 2B** depicts a block diagram illustrating additional components in the session state and/or health manager 401 shown in the example of **FIG. 2A**.

[00142] The client-side components can reside on the mobile device 250 which maintains state information without a need to transmit health-status messages in a network. The mobile device 250 includes a radio, a processing unit, a memory unit having instructions stored thereon which when executed by the processing unit, causes the mobile device to, use a state status of the mobile device to assess if or when communication is expected or predicted to occur at the mobile device. The state status can be used to maintain the session state such that health-status messages need not be transmitted in the network to maintain or convey the session state of mobile device such that resources of the network is conserved.

[00143] In one embodiment, the state status can be determined from information specific to the mobile network to which the mobile device 250 is connected. The state status can also be determined from what applications are accessed via the mobile device 250 or what applications are running or what applications are in the foreground or in the background on the mobile device.

[00144] On the client-side component, the session state and/or health manager 401 can include, for example, a tolerable delay computation module 202 having a delay tracking engine

403, a state map generator 404 and/or a system reset engine 405. Additional or less modules maybe included.

[00145] The session state and/or health manager 401 and its components can alone or in any combination generate a system state map (e.g., by the state map generator 404) which can be used by the distributed proxy system as a mechanism to preserve state or session information between a mobile device and a proxy server. In general, different mobile devices have different state maps, different users using the same device may also have different state maps. Similarly, a given user may have a different or same statement across different devices. A given user may have different state maps during different times of day or days of the week when using a given device. Alternatively, this difference can be reflected in one or more parameters (e.g., time parameter) of a given state map for a user on a given device.

[00146] The state map can be generated by the mobile device 250, and/or it can be generated by the proxy server 325. The state map can also be generated by information aggregated by one or both of the mobile device and the proxy server 325. In general, the state map, once generated, can be optimized by either or both device and server ends of the distributed system based on real-time or near real time performance and dynamic conditions (e.g., network, application status, application host status, etc.).

[00147] Using the state map, or some other representation of expected communications to/from the mobile device 250, the manager 401 can determine a tolerable delay which can elapse before the system suspects that the client or server is down or otherwise not functioning. The delay tracking engine 403 can track the delay. If a max tolerable time of no communications has been reached or exceeded, then the client proxy can automatically poll or request a communication with the server to satisfy a client/application request and/or the client proxy 275 can automatically reset the system (e.g., by the system reset engine 405). Resetting the system can include purging the system cache, or resetting location in the state map. In resetting the system cache, the mobile device 250 can poll the external server such that content requests are satisfied at the mobile device 250.

[00148] Some or all of the above described functionalities can be performed in conjunction with or in lieu of the components on the server side (e.g., in the proxy server 325). **FIG. 3A** depicts a block diagram illustrating an example of server-side components in a distributed proxy and cache system, further including a session state and/or health manager 501 which is

able to convey/manage state among mobile devices and the host server without using heart beat messages. **FIG. 3B** depicts a block diagram illustrating additional components in the session state and/or health manager 501 shown in the example of **FIG. 3A**.

[00149] The server-side components include a system of preserving state information with multiple mobile devices in a wireless network having a proxy server 325 in a wireless network. The proxy server 325 can aggregate state information each of the multiple devices used to determine or predict time frames as to when correspondence or data should be communicated between each of the multiple devices and the proxy server. In one embodiment, the time frame can be represented by a tolerable threshold of lack of communication for each of the multiple devices, and can be specific to each of the multiple devices. The state information can enable the state information to be preserved among the multiple mobile devices and the proxy server 325 the without a need for transmission of messages in the wireless network. In one embodiment, the state information includes, one or more of, what applications are installed on the multiple devices.

[00150] Note that the proxy server 325 can be a distinct from a component of a network operator of the wireless network and in communication with the network operator or the proxy server 325 can be a component of a network operator of the wireless network.

[00151] On the server-side components, the session state and/or health manager 501 can include, for example, a tolerable delay computation module 502 having a delay tracking engine 503, a state map generator 504, a system reset engine 505, and/or a state map tracker and manager 507. Additional or less modules maybe included. The state map tracker and manager 507 is able to track the state maps across multiple mobile devices 250, or across devices 250 in a specific network or devices in a geographic location, or serviced by a network service provider, etc.

[00152] Some or all of the above described functionalities can be performed in conjunction with or in lieu of the components on the client side (e.g., in the local proxy 275).

[00153] **FIG. 4A** depicts a block diagram illustrating an example of client-side components in a distributed proxy and cache system residing on a mobile device (e.g., wireless device) 250 that manages traffic in a wireless network (or broadband network) for resource conservation, content caching, and/or traffic management. The client-side proxy (or local proxy 275) can

further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations.

[00154] The device 250, which can be a portable or mobile device (e.g., any wireless device), such as a portable phone, generally includes, for example, a network interface 208 an operating system 204, a context API 206, and mobile applications which may be proxy-unaware 210 or proxy-aware 220. Note that the device 250 is specifically illustrated in the example of **FIG. 2** as a mobile device, such is not a limitation and that device 250 may be any wireless, broadband, portable/mobile or non-portable device able to receive, transmit signals to satisfy data requests over a network including wired or wireless networks (e.g., Wi-Fi, cellular, Bluetooth, LAN, WAN, etc.).

[00155] The network interface 208 can be a networking module that enables the device 250 to mediate data in a network with an entity that is external to the host server 250, through any known and/or convenient communications protocol supported by the host and the external entity. The network interface 208 can include one or more of a network adaptor card, a wireless network interface card (e.g., SMS interface, Wi-Fi interface, interfaces for various generations of mobile communication standards including but not limited to 2G, 3G, 3.5G, 4G, LTE, etc.), Bluetooth, or whether or not the connection is via a router, an access point, a wireless router, a switch, a multilayer switch, a protocol converter, a gateway, a bridge, a bridge router, a hub, a digital media receiver, and/or a repeater.

[00156] Device 250 can further include client-side components of the distributed proxy and cache system, which can include a local proxy 275 (e.g., a mobile client of a mobile device) and a cache 285. In one embodiment, the local proxy 275 includes a user activity module 215, a proxy API 225, a request/transaction manager 235, a caching policy manager 245 having an application protocol module 248, a traffic shaping engine 255, and/or a connection manager 265. The traffic shaping engine 255 may further include an alignment module 256 and/or a batching module 257, the connection manager 265 may further include a radio controller 266. The request/transaction manager 235 can further include an application behavior detector 236 and/or a prioritization engine 241, the application behavior detector 236 may further include a pattern detector 237 and/or and application profile generator 239. Additional or less components/modules/engines can be included in the local proxy 275 and each illustrated component.

[00157] As used herein, a "module," "a manager," a "handler," a "detector," an "interface," a "controller," a "normalizer," a "generator," an "invalidator," or an "engine" includes a general purpose, dedicated or shared processor and, typically, firmware or software modules that are executed by the processor. Depending upon implementation-specific or other considerations, the module, manager, handler, detector, interface, controller, normalizer, generator, invalidator, or engine can be centralized or its functionality distributed. The module, manager, handler, detector, interface, controller, normalizer, generator, invalidator, or engine can include general or special purpose hardware, firmware, or software embodied in a computer-readable (storage) medium for execution by the processor.

[00158] As used herein, a computer-readable medium or computer-readable storage medium is intended to include all mediums that are statutory (e.g., in the United States, under 35 U.S.C. 101), and to specifically exclude all mediums that are non-statutory in nature to the extent that the exclusion is necessary for a claim that includes the computer-readable (storage) medium to be valid. Known statutory computer-readable mediums include hardware (e.g., registers, random access memory (RAM), non-volatile (NV) storage), but may or may not be limited to hardware.

[00159] In one embodiment, a portion of the distributed proxy and cache system for network traffic management resides in or is in communication with device 250, including local proxy 275 (mobile client) and/or cache 285. The local proxy 275 can provide an interface on the device 250 for users to access device applications and services including email, IM, voice mail, visual voicemail, feeds, Internet, games, productivity tools, or other applications.

[00160] The proxy 275 is generally application independent and can be used by applications (e.g., both proxy-aware and proxy-unaware applications 210 and 220 or mobile applications) to open TCP connections to a remote server (e.g., the server 100 in the examples of **FIG. 1B-1C** and/or server proxy 125/325 shown in the examples of **FIG. 1C** and **FIG. 5A**). In some instances, the local proxy 275 includes a proxy API 225 which can be optionally used to interface with proxy-aware applications 220 (or applications (e.g., mobile applications) on a mobile device (e.g., any wireless device)).

[00161] The applications 210 and 220 can generally include any user application, widgets, software, HTTP-based application, web browsers, video or other multimedia streaming or downloading application, video games, social network applications, email clients, RSS

management applications, application stores, document management applications, productivity enhancement applications, etc. The applications can be provided with the device OS, by the device manufacturer, by the network service provider, downloaded by the user, or provided by others.

[00162] One embodiment of the local proxy 275 includes or is coupled to a context API 206, as shown. The context API 206 may be a part of the operating system 204 or device platform or independent of the operating system 204, as illustrated. The operating system 204 can include any operating system including but not limited to, any previous, current, and/or future versions/releases of, Windows Mobile, iOS, Android, Symbian, Palm OS, Brew MP, Java 2 Micro Edition (J2ME), Blackberry, etc.

[00163] The context API 206 may be a plug-in to the operating system 204 or a particular client/application on the device 250. The context API 206 can detect signals indicative of user or device activity, for example, sensing motion, gesture, device location, changes in device location, device backlight, keystrokes, clicks,, activated touch screen, mouse click or detection of other pointer devices. The context API 206 can be coupled to input devices or sensors on the device 250 to identify these signals. Such signals can generally include input received in response to explicit user input at an input device/mechanism at the device 250 and/or collected from ambient signals/contextual cues detected at or in the vicinity of the device 250 (e.g., light, motion, piezoelectric, etc.).

[00164] In one embodiment, the user activity module 215 interacts with the context API 206 to identify, determine, infer, detect, compute, predict, and/or anticipate, characteristics of user activity on the device 250. Various inputs collected by the context API 206 can be aggregated by the user activity module 215 to generate a profile for characteristics of user activity. Such a profile can be generated by the user activity module 215 with various temporal characteristics. For instance, user activity profile can be generated in real-time for a given instant to provide a view of what the user is doing or not doing at a given time (e.g., defined by a time window, in the last minute, in the last 30 seconds, etc.), a user activity profile can also be generated for a 'session' defined by an application or web page that describes the characteristics of user behavior with respect to a specific task they are engaged in on the device 250, or for a specific time period (e.g., for the last 2 hours, for the last 5 hours).

[00165] Additionally, characteristic profiles can be generated by the user activity module 215 to depict a historical trend for user activity and behavior (e.g., 1 week, 1 mo., 2 mo., etc.). Such historical profiles can also be used to deduce trends of user behavior, for example, access frequency at different times of day, trends for certain days of the week (weekends or week days), user activity trends based on location data (e.g., IP address, GPS, or cell tower coordinate data) or changes in location data (e.g., user activity based on user location, or user activity based on whether the user is on the go, or traveling outside a home region, etc.) to obtain user activity characteristics.

[00166] In one embodiment, user activity module 215 can detect and track user activity with respect to applications, documents, files, windows, icons, and folders on the device 250. For example, the user activity module 215 can detect when an application or window (e.g., a web browser or any other type of application) has been exited, closed, minimized, maximized, opened, moved into the foreground, or into the background, multimedia content playback, etc.

[00167] In one embodiment, characteristics of the user activity on the device 250 can be used to locally adjust behavior of the device (e.g., mobile device or any wireless device) to optimize its resource consumption such as battery/power consumption and more generally, consumption of other device resources including memory, storage, and processing power. In one embodiment, the use of a radio on a device can be adjusted based on characteristics of user behavior (e.g., by the radio controller 266 of the connection manager 265) coupled to the user activity module 215. For example, the radio controller 266 can turn the radio on or off, based on characteristics of the user activity on the device 250. In addition, the radio controller 266 can adjust the power mode of the radio (e.g., to be in a higher power mode or lower power mode) depending on characteristics of user activity.

[00168] In one embodiment, characteristics of the user activity on device 250 can also be used to cause another device (e.g., other computers, a mobile device, a wireless device, or a non-portable device) or server (e.g., host server 100 and 300 in the examples of **FIG. 1B-C** and **FIG. 3A, FIG. 5A**) which can communicate (e.g., via a cellular or other network) with the device 250 to modify its communication frequency with the device 250. The local proxy 275 can use the characteristics information of user behavior determined by the user activity module 215 to instruct the remote device as to how to modulate its communication frequency (e.g., decreasing communication frequency, such as data push frequency if the user is idle,

requesting that the remote device notify the device 250 if new data, changed, data, or data of a certain level of importance becomes available, etc.).

[00169] In one embodiment, the user activity module 215 can, in response to determining that user activity characteristics indicate that a user is active after a period of inactivity, request that a remote device (e.g., server host server 100 and 300 in the examples of **FIG. 1B-C** and **FIG. 3A, 5A**) send the data that was buffered as a result of the previously decreased communication frequency.

[00170] In addition, or in alternative, the local proxy 275 can communicate the characteristics of user activity at the device 250 to the remote device (e.g., host server 100 and 300 in the examples of **FIG. 1B-C** and **FIG. 3A, FIG. 5A)** and the remote device determines how to alter its own communication frequency with the device 250 for network resource conservation and conservation of device 250 resources..

[00171] One embodiment of the local proxy 275 further includes a request/transaction manager 235, which can detect, identify, intercept, process, manage, data requests initiated on the device 250, for example, by applications 210 and/or 220, and/or directly/indirectly by a user request. The request/transaction manager 235 can determine how and when to process a given request or transaction, or a set of requests/transactions, based on transaction characteristics.

[00172] The request/transaction manager 235 can prioritize requests or transactions made by applications and/or users at the device 250, for example by the prioritization engine 241. Importance or priority of requests/transactions can be determined by the request/transaction manager 235 by applying a rule set, for example, according to time sensitivity of the transaction, time sensitivity of the content in the transaction, time criticality of the transaction, time criticality of the data transmitted in the transaction, and/or time criticality or importance of an application making the request.

[00173] In addition, transaction characteristics can also depend on whether the transaction was a result of user-interaction or other user-initiated action on the device (e.g., user interaction with a application (e.g., a mobile application)). In general, a time critical transaction can include a transaction resulting from a user-initiated data transfer, and can be prioritized as such. Transaction characteristics can also depend on the amount of data that will be transferred or is anticipated to be transferred as a result of the requested transaction. For

example, the connection manager 265, can adjust the radio mode (e.g., high power or low power mode via the radio controller 266) based on the amount of data that will need to be transferred.

[00174] In addition, the radio controller 266/connection manager 265 can adjust the radio power mode (high or low) based on time criticality/sensitivity of the transaction. The radio controller 266 can trigger the use of high power radio mode when a time-critical transaction (e.g., a transaction resulting from a user-initiated data transfer, an application running in the foreground, any other event meeting a certain criteria) is initiated or detected.

[00175] In general, the priorities can be set by default, for example, based on device platform, device manufacturer, operating system, etc. Priorities can alternatively or in additionally be set by the particular application; for example, the Facebook application (e.g., a mobile application) can set its own priorities for various transactions (e.g., a status update can be of higher priority than an add friend request or a poke request, a message send request can be of higher priority than a message delete request, for example), an email client or IM chat client may have its own configurations for priority. The prioritization engine 241 may include set of rules for assigning priority.

[00176] The prioritization engine 241 can also track network provider limitations or specifications on application or transaction priority in determining an overall priority status for a request/transaction. Furthermore, priority can in part or in whole be determined by user preferences, either explicit or implicit. A user, can in general, set priorities at different tiers, such as, specific priorities for sessions, or types, or applications (e.g., a browsing session, a gaming session, versus an IM chat session, the user may set a gaming session to always have higher priority than an IM chat session, which may have higher priority than web-browsing session). A user can set application-specific priorities, (e.g., a user may set Facebook-related transactions to have a higher priority than LinkedIn-related transactions), for specific transaction types (e.g., for all send message requests across all applications to have higher priority than message delete requests, for all calendar-related events to have a high priority, etc.), and/or for specific folders.

[00177] The prioritization engine 241 can track and resolve conflicts in priorities set by different entities. For example, manual settings specified by the user may take precedence over device OS settings, network provider parameters/limitations (e.g., set in default for a

network service area, geographic locale, set for a specific time of day, or set based on service/fee type) may limit any user-specified settings and/or application-set priorities. In some instances, a manual synchronization request received from a user can override some, most, or all priority settings in that the requested synchronization is performed when requested, regardless of the individually assigned priority or an overall priority ranking for the requested action.

[00178] Priority can be specified and tracked internally in any known and/or convenient manner, including but not limited to, a binary representation, a multi-valued representation, a graded representation and all are considered to be within the scope of the disclosed technology.

Change (initiated on device)	**Priority**	**Change (initiated on server)**	**Priority**
Send email	High	Receive email	High
Delete email	Low	Edit email	Often not possible to sync (Low if possible)
(Un)read email	Low		
Move message	Low	New email in deleted items	Low
Read more	High		
Download attachment	High	Delete an email	Low
		(Un)Read an email	Low
New Calendar event	High	Move messages	Low
Edit/change Calendar event	High	Any calendar change	High
		Any contact change	High
Add a contact	High	Wipe/lock device	High
Edit a contact	High	Settings change	High
Search contacts	High	Any folder change	High
Change a setting	High	Connector restart	High (if no changes nothing is sent)
Manual send/receive	High		
IM status change	Medium	Social Network Status Updates	Medium
Auction outbid or change notification	High	Sever Weather Alerts	High
Weather Updates	Low	News Updates	Low

Table I

[00179] Table I above shows, for illustration purposes, some examples of transactions with examples of assigned priorities in a binary representation scheme. Additional assignments are possible for additional types of events, requests, transactions, and as previously described, priority assignments can be made at more or less granular levels, e.g., at the session level or at the application level, etc.

[00180] As shown by way of example in the above table, in general, lower priority requests/transactions can include, updating message status as being read, unread, deleting of messages, deletion of contacts; higher priority requests/transactions, can in some instances include, status updates, new IM chat message, new email, calendar event update/cancellation/deletion, an event in a mobile gaming session, or other entertainment related events, a purchase confirmation through a web purchase or online, request to load additional or download content, contact book related events, a transaction to change a device setting, location-aware or location-based events/transactions, or any other events/request/transactions initiated by a user or where the user is known to be, expected to be, or suspected to be waiting for a response, etc.

[00181] Inbox pruning events (e.g., email, or any other types of messages), are generally considered low priority and absent other impending events, generally will not trigger use of the radio on the device 250. Specifically, pruning events to remove old email or other content can be 'piggy backed' with other communications if the radio is not otherwise on, at the time of a scheduled pruning event. For example, if the user has preferences set to 'keep messages for 7 days old,' then instead of powering on the device radio to initiate a message delete from the device 250 the moment that the message has exceeded 7 days old, the message is deleted when the radio is powered on next. If the radio is already on, then pruning may occur as regularly scheduled.

[00182] The request/transaction manager 235, can use the priorities for requests (e.g., by the prioritization engine 241) to manage outgoing traffic from the device 250 for resource optimization (e.g., to utilize the device radio more efficiently for battery conservation). For example, transactions/requests below a certain priority ranking may not trigger use of the radio on the device 250 if the radio is not already switched on, as controlled by the connection manager 265. In contrast, the radio controller 266 can turn on the radio such a request can be sent when a request for a transaction is detected to be over a certain priority level.

[00183] In one embodiment, priority assignments (such as that determined by the local proxy 275 or another device/entity) can be used cause a remote device to modify its communication with the frequency with the mobile device or wireless device. For example, the remote device can be configured to send notifications to the device 250 when data of higher importance is available to be sent to the mobile device or wireless device.

[00184] In one embodiment, transaction priority can be used in conjunction with characteristics of user activity in shaping or managing traffic, for example, by the traffic shaping engine 255. For example, the traffic shaping engine 255 can, in response to detecting that a user is dormant or inactive, wait to send low priority transactions from the device 250, for a period of time. In addition, the traffic shaping engine 255 can allow multiple low priority transactions to accumulate for batch transferring from the device 250 (e.g., via the batching module 257). In one embodiment, the priorities can be set, configured, or readjusted by a user. For example, content depicted in Table I in the same or similar form can be accessible in a user interface on the device 250 and, for example, used by the user to adjust or view the priorities.

[00185] The batching module 257 can initiate batch transfer based on certain criteria. For example, batch transfer (e.g., of multiple occurrences of events, some of which occurred at different instances in time) may occur after a certain number of low priority events have been detected, or after an amount of time elapsed after the first of the low priority event was initiated. In addition, the batching module 257 can initiate batch transfer of the cumulated low priority events when a higher priority event is initiated or detected at the device 250. Batch transfer can otherwise be initiated when radio use is triggered for another reason (e.g., to receive data from a remote device such as host server 100 or 300). In one embodiment, an impending pruning event (pruning of an inbox), or any other low priority events, can be executed when a batch transfer occurs.

[00186] In general, the batching capability can be disabled or enabled at the event/transaction level, application level, or session level, based on any one or combination of the following: user configuration, device limitations/settings, manufacturer specification, network provider parameters/limitations, platform-specific limitations/settings, device OS settings, etc. In one embodiment, batch transfer can be initiated when an application/window/file is closed out, exited, or moved into the background; users can optionally be prompted before initiating a batch transfer; users can also manually trigger batch transfers.

[00187] In one embodiment, the local proxy 275 locally adjusts radio use on the device 250 by caching data in the cache 155. When requests or transactions from the device 250 can be satisfied by content stored in the cache 155, the radio controller 266 need not activate the radio to send the request to a remote entity (e.g., the host server 100, 300, as shown in **FIG. 1B-C** and **FIG. 3A, FIG. 5A** or a content provider/application server such as the server/provider 110 shown in the examples of **FIG. 1B** and **FIG. 1C**). As such, the local proxy 275 can use the local cache 155 and the cache policy manager 245 to locally store data for satisfying data requests to eliminate or reduce the use of the device radio for conservation of network resources and device battery consumption.

[00188] In leveraging the local cache, once the request/transaction manager 225 intercepts a data request by an application on the device 250, the local repository 155 can be queried to determine if there is any locally stored response, and also determine whether the response is valid. When a valid response is available in the local cache 155, the response can be provided to the application on the device 250 without the device 250 needing to access the cellular network or wireless broadband network.

[00189] If a valid response is not available, the local proxy 275 can query a remote proxy (e.g., the server proxy 325 of **FIG. 3A, FIG. 5A**) to determine whether a remotely stored response is valid. If so, the remotely stored response (e.g., which may be stored on the server cache 135 or optional caching server 199 shown in the example of **FIG. 1C**) can be provided to the mobile device, possibly without the mobile device 250 needing to access the cellular network, thus relieving consumption of network resources.

[00190] If a valid cache response is not available, or if cache responses are unavailable for the intercepted data request, the local proxy 275, for example, the caching policy manager 245, can send the data request to a remote proxy (e.g., server proxy 325 of **FIG. 3A, FIG. 5A**) which forwards the data request to a content source (e.g., application server/content provider 110 of **FIG. 1B**) and a response from the content source can be provided through the remote proxy, as will be further described in the description associated with the example host server 300 of **FIG. 3A, FIG. 5A**. The cache policy manager 245 can manage or process requests that use a variety of protocols, including but not limited to HTTP, HTTPS, IMAP, POP, SMTP, XMPP, and/or ActiveSync. The caching policy manager 245 can locally store responses for data requests in the local database 285 as cache entries, for subsequent use in satisfying same or similar data requests.

[00191] The caching policy manager 245 can request that the remote proxy monitor responses for the data request and the remote proxy can notify the device 250 when an unexpected response to the data request is detected. In such an event, the cache policy manager 245 can erase or replace the locally stored response(s) on the device 250 when notified of the unexpected response (e.g., new data, changed data, additional data, etc.) to the data request. In one embodiment, the caching policy manager 245 is able to detect or identify the protocol used for a specific request, including but not limited to HTTP, HTTPS, IMAP, POP, SMTP, XMPP, and/or ActiveSync. In one embodiment, application specific handlers (e.g., via the application protocol module 246 of the caching policy manager 245) on the local proxy 275 allows for optimization of any protocol that can be port mapped to a handler in the distributed proxy (e.g., port mapped on the proxy server 325 in the example of **FIG. 3A, FIG. 5A**).

[00192] In one embodiment, the local proxy 275 notifies the remote proxy such that the remote proxy can monitor responses received for the data request from the content source for changed results prior to returning the result to the device 250, for example, when the data request to the content source has yielded same results to be returned to the mobile device. In general, the local proxy 275 can simulate application server responses for applications on the device 250, using locally cached content. This can prevent utilization of the cellular network for transactions where new/changed data is not available, thus freeing up network resources and preventing network congestion.

[00193] In one embodiment, the local proxy 275 includes an application behavior detector 236 to track, detect, observe, monitor, applications (e.g., proxy-aware and/or unaware applications 210 and 220) accessed or installed on the device 250. Application behaviors, or patterns in detected behaviors (e.g., via the pattern detector 237) of one or more applications accessed on the device 250 can be used by the local proxy 275 to optimize traffic in a wireless network needed to satisfy the data needs of these applications.

[00194] For example, based on detected behavior of multiple applications, the traffic shaping engine 255 can align content requests made by at least some of the applications over the network (wireless network) (e.g., via the alignment module 256). The alignment module 256 can delay or expedite some earlier received requests to achieve alignment. When requests are aligned, the traffic shaping engine 255 can utilize the connection manager to poll over the network to satisfy application data requests. Content requests for multiple applications can be

aligned based on behavior patterns or rules/settings including, for example, content types requested by the multiple applications (audio, video, text, etc.), device (e.g., mobile or wireless device) parameters, and/or network parameters/traffic conditions, network service provider constraints/specifications, etc.

[00195] In one embodiment, the pattern detector 237 can detect recurrences in application requests made by the multiple applications, for example, by tracking patterns in application behavior. A tracked pattern can include, detecting that certain applications, as a background process, poll an application server regularly, at certain times of day, on certain days of the week, periodically in a predictable fashion, with a certain frequency, with a certain frequency in response to a certain type of event, in response to a certain type user query, frequency that requested content is the same, frequency with which a same request is made, interval between requests, applications making a request, or any combination of the above, for example.

[00196] Such recurrences can be used by traffic shaping engine 255 to offload polling of content from a content source (e.g., from an application server/content provider 110 of **FIG. 1B**) that would result from the application requests that would be performed at the mobile device or wireless device 250 to be performed instead, by a proxy server (e.g., proxy server 125 of **FIG. 1C** or proxy server 325 of **FIG. 3A, FIG. 5A**) remote from the device 250. Traffic shaping engine 255 can decide to offload the polling when the recurrences match a rule. For example, there are multiple occurrences or requests for the same resource that have exactly the same content, or returned value, or based on detection of repeatable time periods between requests and responses such as a resource that is requested at specific times during the day. The offloading of the polling can decrease the amount of bandwidth consumption needed by the mobile device 250 to establish a wireless (cellular or other wireless broadband) connection with the content source for repetitive content polls.

[00197] As a result of the offloading of the polling, locally cached content stored in the local cache 155 can be provided to satisfy data requests at the device 250, when content change is not detected in the polling of the content sources. As such, when data has not changed, application data needs can be satisfied without needing to enable radio use or occupying cellular bandwidth in a wireless network. When data has changed and/or new data has been received, the remote entity to which polling is offloaded, can notify the device 250. The remote entity may be the host server 300 as shown in the example of **FIG. 3A, FIG. 5A**.

[00198] In one embodiment, the local proxy 275 can mitigate the need/use of periodic keep-alive messages (heartbeat messages) to maintain TCP/IP connections, which can consume significant amounts of power thus having detrimental impacts on mobile device battery life. The connection manager 265 in the local proxy (e.g., the heartbeat manager 267) can detect, identify, and intercept any or all heartbeat (keep-alive) messages being sent from applications.

[00199] The heartbeat manager 267 can prevent any or all of these heartbeat messages from being sent over the cellular, or other network, and instead rely on the server component of the distributed proxy system (e.g., shown in **FIG. 1C**) to generate the and send the heartbeat messages to maintain a connection with the backend (e.g., application server/provider 110 in the example of **FIG. 1B**).

[00200] The local proxy 275 generally represents any one or a portion of the functions described for the individual managers, modules, and/or engines. The local proxy 275 and device 250 can include additional or less components; more or less functions can be included, in whole or in part, without deviating from the novel art of the disclosure.

[00201] **FIG. 4B** depicts a block diagram illustrating a further example of components in the cache system shown in the example of **FIG. 4A** which is capable of caching and adapting caching strategies for mobile application behavior and/or network conditions.

[00202] In one embodiment, the caching policy manager 245 includes a metadata generator 203, a cache look-up engine 205, a cache appropriateness decision engine 246, a poll schedule generator 247, an application protocol module 248, a cache or connect selection engine 249 and/or a local cache invalidator 244. The cache appropriateness decision engine 246 can further include a timing predictor 246a, a content predictor 246b, a request analyzer 246c, and/or a response analyzer 246d, and the cache or connect selection engine 249 includes a response scheduler 249a. The metadata generator 203 and/or the cache look-up engine 205 are coupled to the cache 285 (or local cache) for modification or addition to cache entries or querying thereof.

[00203] The cache look-up engine 205 may further include an ID or URI filter 205a, the local cache invalidator 244 may further include a TTL manager 244a, and the poll schedule generator 247 may further include a schedule update engine 247a and/or a time adjustment engine 247b. One embodiment of caching policy manager 245 includes an application cache

policy repository 243. In one embodiment, the application behavior detector 236 includes a pattern detector 237, a poll interval detector 238, an application profile generator 239, and/or a priority engine 241. The poll interval detector 238 may further include a long poll detector 238a having a response/request tracking engine 238b. The poll interval detector 238 may further include a long poll hunting detector 238c. The application profile generator 239 can further include a response delay interval tracker 239a.

[00204] The pattern detector 237, application profile generator 239, and the priority engine 241 were also described in association with the description of the pattern detector shown in the example of **FIG. 4A**. One embodiment further includes an application profile repository 242 which can be used by the local proxy 275 to store information or metadata regarding application profiles (e.g., behavior, patterns, type of HTTP requests, etc.)

[00205] The cache appropriateness decision engine 246 can detect, assess, or determine whether content from a content source (e.g., application server/content provider 110 in the example of **FIG. 1C**) with which a mobile device 250 interacts and has content that may be suitable for caching. For example, the decision engine 246 can use information about a request and/or a response received for the request initiated at the mobile device 250 to determine cacheability, potential cacheability, or non-cacheability. In some instances, the decision engine 246 can initially verify whether a request is directed to a blacklisted destination or whether the request itself originates from a blacklisted client or application. If so, additional processing and analysis may not be performed by the decision engine 246 and the request may be allowed to be sent over the air to the server to satisfy the request. The black listed destinations or applications/clients (e.g., mobile applications) can be maintained locally in the local proxy (e.g., in the application profile repository 242) or remotely (e.g., in the proxy server 325 or another entity).

[00206] In one embodiment, the decision engine 246, for example, via the request analyzer 246c, collects information about an application or client request generated at the mobile device 250. The request information can include request characteristics information including, for example, request method. For example, the request method can indicate the type of HTTP request generated by the mobile application or client. In one embodiment, response to a request can be identified as cacheable or potentially cacheable if the request method is a GET request or POST request. Other types of requests (e.g., OPTIONS, HEAD, PUT, DELETE,

TRACE, or CONNECT) may or may not be cached. In general, HTTP requests with uncacheable request methods will not be cached.

[00207] Request characteristics information can further include information regarding request size, for example. Responses to requests (e.g., HTTP requests) with body size exceeding a certain size will not be cached. For example, cacheability can be determined if the information about the request indicates that a request body size of the request does not exceed a certain size. In some instances, the maximum cacheable request body size can be set to 8092 bytes. In other instances, different values may be used, dependent on network capacity or network operator specific settings, for example.

[00208] In some instances, content from a given application server/content provider (e.g., the server/content provider 110 of **FIG. 1C**) is determined to be suitable for caching based on a set of criteria, for example, criteria specifying time criticality of the content that is being requested from the content source. In one embodiment, the local proxy (e.g., the local proxy 175 or 275 of **FIG. 1C** and **FIG. 4A**) applies a selection criteria to store the content from the host server which is requested by an application as cached elements in a local cache on the mobile device to satisfy subsequent requests made by the application.

[00209] The cache appropriateness decision engine 246, further based on detected patterns of requests sent from the mobile device 250 (e.g., by a mobile application or other types of clients on the device 250) and/or patterns of received responses, can detect predictability in requests and/or responses. For example, the request characteristics information collected by the decision engine 246, (e.g., the request analyzer 246c) can further include periodicity information between a request and other requests generated by a same client on the mobile device or other requests directed to the same host (e.g., with similar or same identifier parameters).

[00210] Periodicity can be detected, by the decision engine 246 or the request analyzer 246c, when the request and the other requests generated by the same client occur at a fixed rate or nearly fixed rate, or at a dynamic rate with some identifiable or partially or wholly reproducible changing pattern. If the requests are made with some identifiable pattern (e.g., regular intervals, intervals having a detectable pattern, or trend (e.g., increasing, decreasing, constant, etc.) the timing predictor 246a can determine that the requests made by a given

application on a device is predictable and identify it to be potentially appropriate for caching, at least from a timing standpoint.

[00211] An identifiable pattern or trend can generally include any application or client behavior which may be simulated either locally, for example, on the local proxy 275 on the mobile device 250 or simulated remotely, for example, by the proxy server 325 on the host 300, or a combination of local and remote simulation to emulate application behavior.

[00212] In one embodiment, the decision engine 246, for example, via the response analyzer 246d, can collect information about a response to an application or client request generated at the mobile device 250. The response is typically received from a server or the host of the application (e.g., mobile application) or client which sent the request at the mobile device 250. In some instances, the mobile client or application can be the mobile version of an application (e.g., social networking, search, travel management, voicemail, contact manager, email) or a web site accessed via a web browser or via a desktop client.

[00213] For example, response characteristics information can include an indication of whether transfer encoding or chunked transfer encoding is used in sending the response. In some instances, responses to HTTP requests with transfer encoding or chunked transfer encoding are not cached, and therefore are also removed from further analysis. The rationale here is that chunked responses are usually large and non-optimal for caching, since the processing of these transactions may likely slow down the overall performance. Therefore, in one embodiment, cacheability or potential for cacheability can be determined when transfer encoding is not used in sending the response.

[00214] In addition, the response characteristics information can include an associated status code of the response which can be identified by the response analyzer 246d. In some instances, HTTP responses with uncacheable status codes are typically not cached. The response analyzer 246d can extract the status code from the response and determine whether it matches a status code which is cacheable or uncacheable. Some cacheable status codes include by way of example: 200-OK, 301-Redirect, 302-Found, 303-See other, 304 - Not Modified, 307 Temporary Redirect, or 500 – Internal server error. Some uncacheable status codes can include, for example, 403 – Forbidden or 404 – Not found.

[00215] In one embodiment, cacheability or potential for cacheability can be determined if the information about the response does not indicate an uncacheable status code or indicates a

cacheable status code. If the response analyzer 246d detects an uncacheable status code associated with a given response, the specific transaction (request/response pair) may be eliminated from further processing and determined to be uncacheable on a temporary basis, a semi-permanent, or a permanent basis. If the status code indicates cacheability, the transaction (e.g., request and/or response pair) may be subject to further processing and analysis to confirm cacheability, as shown in the example flow charts of **FIG. 9-13**.

[00216] Response characteristics information can also include response size information. In general, responses can be cached locally at the mobile device 250 if the responses do not exceed a certain size. In some instances, the default maximum cached response size is set to 115 KB. In other instances, the max cacheable response size may be different and/or dynamically adjusted based on operating conditions, network conditions, network capacity, user preferences, network operator requirements, or other application-specific, user specific, and/or device-specific reasons. In one embodiment, the response analyzer 246d can identify the size of the response, and cacheability or potential for cacheability can be determined if a given threshold or max value is not exceeded by the response size.

[00217] Furthermore, response characteristics information can include response body information for the response to the request and other response to other requests generated by a same client on the mobile device, or directed to a same content host or application server. The response body information for the response and the other responses can be compared, for example, by the response analyzer 246d, to prevent the caching of dynamic content (or responses with content that changes frequently and cannot be efficiently served with cache entries, such as financial data, stock quotes, news feeds, real-time sporting event activities, etc.), such as content that would no longer be relevant or up-to-date if served from cached entries.

[00218] The cache appropriateness decision engine 246 (e.g., the content predictor 246b) can definitively identify repeatability or identify indications of repeatability, potential repeatability, or predictability in responses received from a content source (e.g., the content host/application server 110 shown in the example of **FIG. 1B-C**). Repeatability can be detected by, for example, tracking at least two responses received from the content source and determines if the two responses are the same. For example, cacheability can be determined, by the response analyzer 246d, if the response body information for the response and the other responses sent by the same mobile client or directed to the same host/server are same or

substantially the same. The two responses may or may not be responses sent in response to consecutive requests. In one embodiment, hash values of the responses received for requests from a given application are used to determine repeatability of content (with or without heuristics) for the application in general and/or for the specific request. Additional same responses may be required for some applications or under certain circumstances.

[00219] Repeatability in received content need not be 100% ascertained. For example, responses can be determined to be repeatable if a certain number or a certain percentage of responses are the same, or similar. The certain number or certain percentage of same/similar responses can be tracked over a select period of time, set by default or set based on the application generating the requests (e.g., whether the application is highly dynamic with constant updates or less dynamic with infrequent updates). Any indicated predictability or repeatability, or possible repeatability, can be utilized by the distributed system in caching content to be provided to a requesting application or client on the mobile device 250.

[00220] In one embodiment, for a long poll type request, the local proxy 175 can begin to cache responses on a third request when the response delay times for the first two responses are the same, substantially the same, or detected to be increasing in intervals. In general, the received responses for the first two responses should be the same, and upon verifying that the third response received for the third request is the same (e.g., if R0 = R1 = R2), the third response can be locally cached on the mobile device. Less or more same responses may be required to begin caching, depending on the type of application, type of data, type of content, user preferences, or carrier/network operator specifications.

[00221] Increasing response delays with same responses for long polls can indicate a hunting period (e.g., a period in which the application/client on the mobile device is seeking the longest time between a request and response that a given network will allow as detected by the long poll hunting detector 238c of the application behavior detector 236.

[00222] An example can be described below using T0, T1, T2, where T indicates the delay time between when a request is sent and when a response (e.g., the response header) is detected/received for consecutive requests:

T0 = Response0(t) – Request0(t) = 180 s. (+/- tolerance)
T1 = Response1(t) – Request1(t) = 240 s. (+/- tolerance)
T2 = Response2(t) – Request2(t) = 500 s. (+/- tolerance)

[00223] In the example timing sequence shown above, T0 < T1 < T2, this may indicate a hunting pattern for a long poll when network timeout has not yet been reached or exceeded. Furthermore, if the responses R0, R1, and R2 received for the three requests are the same, R2 can be cached. In this example, R2 is cached during the long poll hunting period without waiting for the long poll to settle, thus expediting response caching (e.g., this is optional accelerated caching behavior which can be implemented for all or select applications).

[00224] As such, the local proxy 275 can specify information that can be extracted from the timing sequence shown above (e.g., polling schedule, polling interval, polling type) to the proxy server and begin caching and to request the proxy server to begin polling and monitoring the source (e.g., using any of T0, T1, T2 as polling intervals but typically T2, or the largest detected interval without timing out, and for which responses from the source is received will be sent to the proxy server 325 of **FIG. 5A** for use in polling the content source (e.g., application server/service provider 310)).

[00225] However, if the time intervals are detected to be getting shorter, the application (e.g., mobile application)/client may still be hunting for a time interval for which a response can be reliably received from the content source (e.g., application/server server/provider 110 or 310), and as such caching typically should not begin until the request/response intervals indicate the same time interval or an increasing time interval, for example, for a long poll type request.

[00226] An example of handling a detected decreasing delay can be described below using T0, T1, T2, T3, and T4 where T indicates the delay time between when a request is sent and when a response (e.g., the response header) is detected/received for consecutive requests:

T0 = Response0(t) – Request0(t) = 160 s. (+/- tolerance)
T1 = Response1(t) – Request1(t) = 240 s. (+/- tolerance)
T2 = Response2(t) – Request2(t) = 500 s. (+/- tolerance)
T3 = Time out at 700 s. (+/- tolerance)
T4 = Response4(t) – Request4(t) = 600 (+/- tolerance)

[00227] If a pattern for response delays T1 < T2 < T3 > T4 is detected, as shown in the above timing sequence (e.g., detected by the long poll hunting detector 238c of the application

behavior detector 236), it can be determined that T3 likely exceeded the network time out during a long poll hunting period. In Request 3, a response likely was not received since the connection was terminated by the network, application, server, or other reason before a response was sent or available. On Request 4 (after T4), if a response (e.g., Response 4) is detected or received, the local proxy 275 can then use the response for caching (if the content repeatability condition is met). The local proxy can also use T4 as the poll interval in the polling schedule set for the proxy server to monitor/poll the content source.

[00228] Note that the above description shows that caching can begin while long polls are in hunting mode in the event of detecting increasing response delays, as long as responses are received and not timed out for a given request. This can be referred to as the optional accelerated caching during long poll hunting. Caching can also begin after the hunting mode (e.g., after the poll requests have settled to a constant or near constant delay value) has completed. Note that hunting may or may not occur for long polls and when hunting occurs; the proxy 275 can generally detect this and determine whether to begin to cache during the hunting period (increasing intervals with same responses) or wait until the hunt settles to a stable value.

[00229] In one embodiment, the timing predictor 246a of the cache appropriateness decision engine 246 can track timing of responses received from outgoing requests from an application (e.g., mobile application) or client to detect any identifiable patterns which can be partially wholly reproducible, such that locally cached responses can be provided to the requesting client on the mobile device 250 in a manner that simulates content source (e.g., application server/content provider 110 or 310) behavior. For example, the manner in which (e.g., from a timing standpoint) responses or content would be delivered to the requesting application/client on the device 250. This ensures preservation of user experience when responses to application or mobile client requests are served from a local and/or remote cache instead of being retrieved/received directly from the content source (e.g., application, content provider 110 or 310).

[00230] In one embodiment, the decision engine 246 or the timing predictor 246a determines the timing characteristics a given application (e.g., mobile application) or client from, for example, the request/response tracking engine 238b and/or the application profile generator 239 (e.g., the response delay interval tracker 239a). Using the timing characteristics, the timing predictor 246a determines whether the content received in response to the requests

are suitable or are potentially suitable for caching. For example, poll request intervals between two consecutive requests from a given application can be used to determine whether request intervals are repeatable (e.g., constant, near constant, increasing with a pattern, decreasing with a pattern, etc.) and can be predicted and thus reproduced at least some of the times either exactly or approximated within a tolerance level.

[00231] In some instances, the timing characteristics of a given request type for a specific application, for multiple requests of an application, or for multiple applications can be stored in the application profile repository 242. The application profile repository 242 can generally store any type of information or metadata regarding application request/response characteristics including timing patterns, timing repeatability, content repeatability, etc.

[00232] The application profile repository 242 can also store metadata indicating the type of request used by a given application (e.g., long polls, long-held HTTP requests, HTTP streaming, push, COMET push, etc.). Application profiles indicating request type by applications can be used when subsequent same/similar requests are detected, or when requests are detected from an application which has already been categorized. In this manner, timing characteristics for the given request type or for requests of a specific application which has been tracked and/or analyzed, need not be reanalyzed.

[00233] Application profiles can be associated with a time-to-live (e.g., or a default expiration time). The use of an expiration time for application profiles, or for various aspects of an application or request's profile can be used on a case by case basis. The time-to-live or actual expiration time of application profile entries can be set to a default value or determined individually, or a combination thereof. Application profiles can also be specific to wireless networks, physical networks, network operators, or specific carriers.

[00234] One embodiment includes an application blacklist manager 201. The application blacklist manager 201 can be coupled to the application cache policy repository 243 and can be partially or wholly internal to local proxy or the caching policy manager 245. Similarly, the blacklist manager 201 can be partially or wholly internal to local proxy or the application behavior detector 236. The blacklist manager 201 can aggregate, track, update, manage, adjust, or dynamically monitor a list of destinations of servers/host that are 'blacklisted,' or identified as not cached, on a permanent or temporary basis. The blacklist of destinations, when identified in a request, can potentially be used to allow the request to be sent over the

(cellular) network for servicing. Additional processing on the request may not be performed since it is detected to be directed to a blacklisted destination.

[00235] Blacklisted destinations can be identified in the application cache policy repository 243 by address identifiers including specific URIs or patterns of identifiers including URI patterns. In general, blacklisted destinations can be set by or modified for any reason by any party including the user (owner/user of mobile device 250), operating system/mobile platform of device 250, the destination itself, network operator (of cellular network), Internet service provider, other third parties, or according to a list of destinations for applications known to be uncacheable/not suited for caching. Some entries in the blacklisted destinations may include destinations aggregated based on the analysis or processing performed by the local proxy (e.g., cache appropriateness decision engine 246).

[00236] For example, applications or mobile clients on the mobile device for which responses have been identified as non-suitable for caching can be added to the blacklist. Their corresponding hosts/servers may be added in addition to or in lieu of an identification of the requesting application/client on the mobile device 250. Some or all of such clients identified by the proxy system can be added to the blacklist. For example, for all application clients or applications that are temporarily identified as not being suitable for caching, only those with certain detected characteristics (based on timing, periodicity, frequency of response content change, content predictability, size, etc.) can be blacklisted.

[00237] The blacklisted entries may include a list of requesting applications or requesting clients on the mobile device (rather than destinations) such that, when a request is detected from a given application or given client, it may be sent through the network for a response, since responses for blacklisted clients/applications are in most circumstances not cached.

[00238] A given application profile may also be treated or processed differently (e.g., different behavior of the local proxy 275 and the remote proxy 325) depending on the mobile account associated with a mobile device from which the application is being accessed. For example, a higher paying account, or a premier account may allow more frequent access of the wireless network or higher bandwidth allowance thus affecting the caching policies implemented between the local proxy 275 and proxy server 325 with an emphasis on better performance compared to conservation of resources. A given application profile may also be

treated or processed differently under different wireless network conditions (e.g., based on congestion or network outage, etc.).

[00239] Note that cache appropriateness can be determined, tracked, and managed for multiple clients or applications on the mobile device 250. Cache appropriateness can also be determined for different requests or request types initiated by a given client or application on the mobile device 250. The caching policy manager 245, along with the timing predictor 246a and/or the content predictor 246b which heuristically determines or estimates predictability or potential predictability, can track, manage and store cacheability information for various application or various requests for a given application. Cacheability information may also include conditions (e.g., an application can be cached at certain times of the day, or certain days of the week, or certain requests of a given application can be cached, or all requests with a given destination address can be cached) under which caching is appropriate which can be determined and/or tracked by the cache appropriateness decision engine 246 and stored and/or updated when appropriate in the application cache policy repository 243 coupled to the cache appropriateness decision engine 246.

[00240] The information in the application cache policy repository 243 regarding cacheability of requests, applications, and/or associated conditions can be used later on when same requests are detected. In this manner, the decision engine 246 and/or the timing and content predictors 246a/b need not track and reanalyze request/response timing and content characteristics to make an assessment regarding cacheability. In addition, the cacheability information can in some instances be shared with local proxies of other mobile devices by way of direct communication or via the host server (e.g., proxy server 325 of host server 300).

[00241] For example, cacheability information detected by the local proxy 275 on various mobile devices can be sent to a remote host server or a proxy server 325 on the host server (e.g., host server 300 or proxy server 325 shown in the example of **FIG. 5A**, host 100 and proxy server 125 in the example of **FIG. 1B-C**). The remote host or proxy server can then distribute the information regarding application-specific, request-specific cacheability information and/or any associated conditions to various mobile devices or their local proxies in a wireless network or across multiple wireless networks (same service provider or multiple wireless service providers) for their use.

[00242] In general, the selection criteria for caching can further include, by way of example but not limitation, the state of the mobile device indicating whether the mobile device is active or inactive, network conditions, and/or radio coverage statistics. The cache appropriateness decision engine 246 can in any one or any combination of the criteria, and in any order, identifying sources for which caching may be suitable.

[00243] Once application servers/content providers having identified or detected content that is potentially suitable for local caching on the mobile device 250, the cache policy manager 245 can proceed to cache the associated content received from the identified sources by storing content received from the content source as cache elements in a local cache (e.g., local cache 185 or 285 shown in the examples of **FIG. 1C** and **FIG. 4A**, respectively) on the mobile device 250.

[00244] The response can be stored in the cache 285 (e.g., also referred as the local cache) as a cache entry. In addition to the response to a request, the cached entry can include response metadata having additional information regarding caching of the response. The metadata may be generated by the metadata generator 203 and can include, for example, timing data such as the access time of the cache entry or creation time of the cache entry. Metadata can include additional information, such as any information suited for use in determining whether the response stored as the cached entry is used to satisfy the subsequent response. For example, metadata information can further include, request timing history (e.g., including request time, request start time, request end time), hash of the request and/or response, time intervals or changes in time intervals, etc.

[00245] The cache entry is typically stored in the cache 285 in association with a time-to-live (TTL), which for example may be assigned or determined by the TTL manager 244a of the cache invalidator 244. The time-to-live of a cache entry is the amount of time the entry is persisted in the cache 285 regardless of whether the response is still valid or relevant for a given request or client/application on the mobile device 250. For example, if the time-to-live of a given cache entry is set to 12 hours, the cache entry is purged, removed, or otherwise indicated as having exceeded the time-to-live, even if the response body contained in the cache entry is still current and applicable for the associated request.

[00246] A default time-to-live can be automatically used for all entries unless otherwise specified (e.g., by the TTL manager 244a), or each cache entry can be created with its

individual TTL (e.g., determined by the TTL manager 244a based on various dynamic or static criteria). Note that each entry can have a single time-to-live associated with both the response data and any associated metadata. In some instances, the associated metadata may have a different time-to-live (e.g., a longer time-to-live) than the response data.

[00247] The content source having content for caching can, in addition or in alternate, be identified to a proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 5A**, respectively) remote from and in wireless communication with the mobile device 250 such that the proxy server can monitor the content source (e.g., application server/content provider 110) for new or changed data. Similarly, the local proxy (e.g., the local proxy 175 or 275 of **FIG. 1C** and **FIG. 4A**, respectively) can identify to the proxy server that content received from a specific application server/content provider is being stored as cached elements in the local cache 285.

[00248] Once content has been locally cached, the cache policy manager 245, upon receiving future polling requests to contact the application server/content host (e.g., 110 or 310), can retrieve the cached elements from the local cache to respond to the polling request made at the mobile device 250 such that a radio of the mobile device is not activated to service the polling request. For example, the cache look-up engine 205 can query the cache 285 to identify the response to be served to a response. The response can be served from the cache in response to identifying a matching cache entry and also using any metadata stored with the response in the cache entry. The cache entries can be queried by the cache look-up engine using a URI of the request or another type of identifier (e.g., via the ID or URI filter 205a). The cache-lookup engine 205 can further use the metadata (e.g., extract any timing information or other relevant information) stored with the matching cache entry to determine whether response is still suited for use in being served to a current request.

[00249] Note that the cache-look-up can be performed by the engine 205 using one or more of various multiple strategies. In one embodiment, multiple cook-up strategies can be executed sequentially on each entry store din the cache 285, until at least one strategy identifies a matching cache entry. The strategy employed to performing cache look-up can include a strict matching criteria or a matching criteria which allows for non-matching parameters.

[00250] For example, the look-up engine 205 can perform a strict matching strategy which searches for an exact match between an identifier (e.g., a URI for a host or resource)

referenced in a present request for which the proxy is attempting to identify a cache entry and an identifier stored with the cache entries. In the case where identifiers include URIs or URLs, the matching algorithm for strict matching will search for a cache entry where all the parameters in the URLs match. For example:

Example 1.

1. Cache contains entry for http://test.com/products/
2. Request is being made to URI http://test.com/products/

Strict strategy will find a match, since both URIs are same.

Example 2.

1. Cache contains entry for http://test.com/products/?query=all
2. Request is being made to URI http://test.com/products/?query=sub

[00251] Under the strict strategy outlined above, a match will not be found since the URIs differ in the query parameter.

[00252] In another example strategy, the look-up engine 205 looks for a cache entry with an identifier that partially matches the identifier references in a present request for which the proxy is attempting to identify a matching cache entry. For example, the look-up engine 205 may look for a cache entry with an identifier which differs from the request identifier by a query parameter value. In utilizing this strategy, the look-up engine 205 can collect information collected for multiple previous requests (e.g., a list of arbitrary parameters in an identifier) to be later checked with the detected arbitrary parameter in the current request. For example, in the case where cache entries are stored with URI or URL identifiers, the look-up engine searches for a cache entry with a URI differing by a query parameter. If found, the engine 205 can examine the cache entry for information collected during previous requests (e.g., a list of arbitrary parameters) and checked whether the arbitrary parameter detected in or extracted from the current URI/URL belongs to the arbitrary parameters list.

Example 1.

1. Cache contains entry for http://test.com/products/?query=all, where query is marked as arbitrary.
2. Request is being made to URI http://text.com/products/?query=sub

Match will be found, since query parameter is marked as arbitrary.

Example 2.

1. Cache contains entry for http://test.com/products/?query=all, where query is marked as arbitrary.
2. Request is being made to URI http://test.com/products/?query=sub&sort=asc

Match will not be found, since current request contains sort parameter which is not marked as arbitrary in the cache entry.

[00253] Additional strategies for detecting cache hit may be employed. These strategies can be implemented singly or in any combination thereof. A cache-hit can be determined when any one of these strategies determines a match. A cache miss may be indicated when the look-up engine 205 determines that the requested data cannot be served from the cache 285, for any reason. For example, a cache miss may be determined when no cache entries are identified for any or all utilized look-up strategies.

[00254] Cache miss may also be determined when a matching cache entry exists but determined to be invalid or irrelevant for the current request. For example, the look-up engine 205 may further analyze metadata (e.g., which may include timing data of the cache entry) associated with the matching cache entry to determine whether it is still suitable for use in responding to the present request.

[00255] When the look-up engine 205 has identified a cache hit (e.g., an event indicating that the requested data can be served from the cache), the stored response in the matching cache entry can be served from the cache to satisfy the request of an application/client.

[00256] By servicing requests using cache entries stored in cache 285, network bandwidth and other resources need not be used to request/receive poll responses which may have not changed from a response that has already been received at the mobile device 250. Such servicing and fulfilling application (e.g., mobile application) requests locally via cache entries in the local cache 285 allows for more efficient resource and mobile network traffic utilization and management since the request need not be sent over the wireless network further consuming bandwidth. In general, the cache 285 can be persisted between power on/off of the mobile device 250, and persisted across application/client refreshes and restarts.

[00257] For example, the local proxy 275, upon receipt of an outgoing request from its mobile device 250 or from an application or other type of client on the mobile device 250, can

intercept the request and determine whether a cached response is available in the local cache 285 of the mobile device 250. If so, the outgoing request is responded to by the local proxy 275 using the cached response on the cache of the mobile device. As such, the outgoing request can be filled or satisfied without a need to send the outgoing request over the wireless network, thus conserving network resources and battery consumption.

[00258] In one embodiment, the responding to the requesting application/client on the device 250 is timed to correspond to a manner in which the content server would have responded to the outgoing request over a persistent connection (e.g., over the persistent connection, or long-held HTTP connection, long poll type connection, that would have been established absent interception by the local proxy). The timing of the response can be emulated or simulated by the local proxy 275 to preserve application behavior such that end user experience is not affected, or minimally affected by serving stored content from the local cache 285 rather than fresh content received from the intended content source (e.g., content host/application server 110 of **FIG. 1B-C**). The timing can be replicated exactly or estimated within a tolerance parameter, which may go unnoticed by the user or treated similarly by the application so as to not cause operation issues.

[00259] For example, the outgoing request can be a request for a persistent connection intended for the content server (e.g., application server/content provider of examples of **FIG. 1B-1C**). In a persistent connection (e.g., long poll, COMET-style push or any other push simulation in asynchronous HTTP requests, long-held HTTP request, HTTP streaming, or others) with a content source (server), the connection is held for some time after a request is sent. The connection can typically be persisted between the mobile device and the server until content is available at the server to be sent to the mobile device. Thus, there typically can be some delay in time between when a long poll request is sent and when a response is received from the content source. If a response is not provided by the content source for a certain amount of time, the connection may also terminate due to network reasons (e.g., socket closure) if a response is not sent.

[00260] Thus, to emulate a response from a content server sent over a persistent connection (e.g., a long poll style connection), the manner of response of the content server can be simulated by allowing a time interval to elapse before responding to the outgoing request with the cached response. The length of the time interval can be determined on a request by request basis or on an application by application (client by client basis), for example.

[00261] In one embodiment, the time interval is determined based on request characteristics (e.g., timing characteristics) of an application on the mobile device from which the outgoing request originates. For example, poll request intervals (e.g., which can be tracked, detected, and determined by the long poll detector 238a of the poll interval detector 238) can be used to determine the time interval to wait before responding to a request with a local cache entry and managed by the response scheduler 249a.

[00262] One embodiment of the cache policy manager 245 includes a poll schedule generator 247 which can generate a polling schedule for one or more applications on the mobile device 250. The polling schedule can specify a polling interval that can be employed by an entity which is physically distinct and/or separate from the mobile device 250 in monitoring the content source for one or more applications (such that cached responses can be verified periodically by polling a host server (host server 110 or 310) to which the request is directed) on behalf of the mobile device. One example of such an external entity which can monitor the content at the source for the mobile device 250 is a proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 5A-D**).

[00263] The polling schedule (e.g., including a rate/frequency of polling) can be determined, for example, based on the interval between the polling requests directed to the content source from the mobile device. The polling schedule or rate of polling may be determined at the mobile device 250 (by the local proxy). In one embodiment, the poll interval detector 238 of the application behavior detector 236 can monitor polling requests directed to a content source from the mobile device 250 in order to determine an interval between the polling requests made from any or all application (e.g., mobile application).

[00264] For example, the poll interval detector 238 can track requests and responses for applications or clients on the device 250. In one embodiment, consecutive requests are tracked prior to detection of an outgoing request initiated from the application (e.g., mobile application) on the mobile device 250 by the same mobile client or application (e.g., mobile application). The polling rate can be determined using request information collected for the request for which the response is cached. In one embodiment, the rate is determined from averages of time intervals between previous requests generated by the same client which generated the request. For example, a first interval may be computed between the current request and a previous request, and a second interval can be computed between the two

previous requests. The polling rate can be set from the average of the first interval and the second interval and sent to the proxy server in setting up the caching strategy.

[00265] Alternate intervals may be computed in generating an average; for example, multiple previous requests in addition to two previous requests may be used, and more than two intervals may be used in computing an average. In general, in computing intervals, a given request need not have resulted in a response to be received from the host server/content source in order to use it for interval computation. In other words, the timing characteristics of a given request may be used in interval computation, as long as the request has been detected, even if the request failed in sending, or if the response retrieval failed.

[00266] One embodiment of the poll schedule generator 247 includes a schedule update engine 247a and/or a time adjustment engine 247b. The schedule update engine 247a can determine a need to update a rate or polling interval with which a given application server/content host from a previously set value, based on a detected interval change in the actual requests generated from a client or application (e.g., mobile application) on the mobile device 250.

[00267] For example, a request for which a monitoring rate was determined may now be sent from the application (e.g., mobile application) or client at a different request interval. The scheduled update engine 247a can determine the updated polling interval of the actual requests and generate a new rate, different from the previously set rate to poll the host at on behalf of the mobile device 250. The updated polling rate can be communicated to the remote proxy (proxy server 325) over the cellular network for the remote proxy to monitor the given host. In some instances, the updated polling rate may be determined at the remote proxy or remote entity which monitors the host.

[00268] In one embodiment, the time adjustment engine 247b can further optimize the poll schedule generated to monitor the application server/content source (110 or 310). For example, the time adjustment engine 247b can optionally specify a time to start polling to the proxy server. For example, in addition to setting the polling interval at which the proxy server is to monitor the application, server/content host can also specify the time at which an actual request was generated at the mobile client/application.

[00269] However, in some cases, due to inherent transmission delay or added network delays or other types of latencies, the remote proxy server receives the poll setup from the local

proxy with some delay (e.g., a few minutes, or a few seconds). This has the effect of detecting response change at the source after a request is generated by the mobile client/application causing the invalidate of the cached response to occur after it has once again been served to the application after the response is no longer current or valid. This discrepancy is further illustrated diagrammatically in the data timing diagram of **FIG. 21**.

[00270] To resolve this non-optimal result of serving the out-dated content once again before invalidating it, the time adjustment engine 247b can specify the time (t0) at which polling should begin in addition to the rate, where the specified initial time t0 can be specified to the proxy server 325 as a time that is less than the actual time when the request was generated by the mobile app/client. This way, the server polls the resource slightly before the generation of an actual request by the mobile client such that any content change can be detected prior to an actual application request. This prevents invalid or irrelevant out-dated content/response from being served once again before fresh content is served.

[00271] In one embodiment, an outgoing request from a mobile device 250 is detected to be for a persistent connection (e.g., a long poll, COMET style push, and long-held (HTTP) request) based on timing characteristics of prior requests from the same application or client on the mobile device 250. For example, requests and/or corresponding responses can be tracked by the request/response tracking engine 238b of the long poll detector 238a of the poll interval detector 238.

[00272] The timing characteristics of the consecutive requests can be determined to set up a polling schedule for the application or client. The polling schedule can be used to monitor the content source (content source/application server) for content changes such that cached content stored on the local cache in the mobile device 250 can be appropriately managed (e.g., updated or discarded). In one embodiment, the timing characteristics can include, for example, a response delay time ('D') and/or an idle time ('IT').

[00273] In one embodiment, the response/request tracking engine 238b can track requests and responses to determine, compute, and/or estimate, the timing diagrams for applicant or client requests.

[00274] For example, the response/request tracking engine 238b detects a first request (Request 0) initiated by a client on the mobile device and a second request (Request 1) initiated

by the client on the mobile device after a response is received at the mobile device responsive to the first request. The second request is one that is subsequent to the first request.

[00275] In one embodiment, the response/request tracking engine 238b can track requests and responses to determine, compute, and/or estimate the timing diagrams for applicant or client requests. The response/request tracking engine 238b can detect a first request initiated by a client on the mobile device and a second request initiated by the client on the mobile device after a response is received at the mobile device responsive to the first request. The second request is one that is subsequent to the first request.

[00276] The response/request tracking engine 238b further determines relative timings between the first, second requests, and the response received in response to the first request. In general, the relative timings can be used by the long poll detector 238a to determine whether requests generated by the application are long poll requests.

[00277] Note that in general, the first and second requests that are used by the response/request tracking engine 238b in computing the relative timings are selected for use after a long poll hunting period has settled or in the event when long poll hunting does not occur. Timing characteristics that are typical of a long poll hunting period and can be, for example, detected by the long poll hunting detector 238c. In other words, the requests tracked by the response/request tracking engine 238b and used for determining whether a given request is a long poll occurs after the long poll has settled (.

[00278] In one embodiment, the long poll hunting detector 238c can identify or detect hunting mode, by identifying increasing request intervals (e.g., increasing delays). The long poll hunting detector 238a can also detect hunting mode by detecting increasing request intervals, followed by a request with no response (e.g., connection timed out), or by detecting increasing request intervals followed by a decrease in the interval. In addition, the long poll hunting detector 238c can apply a filter value or a threshold value to request-response time delay value (e.g., an absolute value) above which the detected delay can be considered to be a long poll request-response delay. The filter value can be any suitable value characteristic of long polls and/or network conditions (e.g., 2s, 5s, 10s, 15s, 20s, etc.) and can be used as a filter or threshold value.

[00279] The response delay time ('D') refers to the start time to receive a response after a request has been sent and the idle refers to time to send a subsequent request after the response

has been received. In one embodiment, the outgoing request is detected to be for a persistent connection based on a comparison (e.g., performed by the tracking engine 238b) of the response delay time relative ('D') or average of ('D') (e.g., any average over any period of time) to the idle time ('IT'), for example, by the long poll detector 238a. The number of averages used can be fixed, dynamically adjusted, or changed over a longer period of time. For example, the requests initiated by the client are determined to be long poll requests if the response delay time interval is greater than the idle time interval (D >IT or D>>IT). In one embodiment, the tracking engine 238b of the long poll detector computes, determines, or estimates the response delay time interval as the amount of time elapsed between time of the first request and initial detection or full receipt of the response.

[00280] In one embodiment, a request is detected to be for a persistent connection when the idle time ('IT') is short since persistent connections, established in response to long poll requests or long poll HTTP requests for example, can also be characterized in detecting immediate or near-immediate issuance of a subsequent request after receipt of a response to a previous request (e.g., IT ~0). As such, the idle time ('IT') can also be used to detect such immediate or near-immediate re-request to identify long poll requests. The absolute or relative timings determined by the tracking engine 238b are used to determine whether the second request is immediately or near-immediately re-requested after the response to the first request is received. For example, a request may be categorized as a long poll request if D + RT + IT ~ D + RT since IT is small for this to hold true. IT may be determined to be small if it is less than a threshold value. Note that the threshold value could be fixed or calculated over a limited time period (a session, a day, a month, etc.), or calculated over a longer time period (e.g., several months or the life of the analysis). For example, for every request, the average IT can be determined, and the threshold can be determined using this average IT (e.g., the average IT less a certain percentage may be used as the threshold). This can allow the threshold to automatically adapt over time to network conditions and changes in server capability, resource availability or server response. A fixed threshold can take upon any value including by way of example but not limitation (e.g., 1s, 2s, 3s, etc.).

[00281] In one embodiment, the long poll detector 238a can compare the relative timings (e.g., determined by the tracker engine 238b) to request-response timing characteristics for other applications to determine whether the requests of the application are long poll requests. For example, the requests initiated by a client or application can be determined to be long poll

requests if the response delay interval time ('D') or the average response delay interval time (e.g., averaged over x number of requests or any number of delay interval times averaged over x amount of time) is greater than a threshold value.

[00282] The threshold value can be determined using response delay interval times for requests generated by other clients, for example by the request/response tracking engine 238b and/or by the application profile generator 239 (e.g., the response delay interval tracker 239a). The other clients may reside on the same mobile device and the threshold value is determined locally by components on the mobile device. The threshold value can be determined for all requests over all resources server over all networks, for example. The threshold value can be set to a specific constant value (e.g., 30 seconds, for example) to be used for all requests, or any request which does not have an applicable threshold value (e.g., long poll is detected if D > 30 seconds).

[00283] In some instances, the other clients reside on different mobile devices and the threshold can be determined by a proxy server (e.g., proxy server 325 of the host 300 shown in the example of **FIG. 5A-B**) which is external to the mobile device and able to communicate over a wireless network with the multiple different mobile devices, as will be further described with reference to **FIG. 5B**.

[00284] In one embodiment, the cache policy manager 245 sends the polling schedule to the proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 5A**) and can be used by the proxy server in monitoring the content source, for example, for changed or new content (updated response different from the cached response associated with a request or application). A polling schedule sent to the proxy can include multiple timing parameters including but not limited to interval (time from request 1 to request 2) or a time out interval (time to wait for response, used in long polls, for example).

[00285] For example, in the case when the local proxy 275 detects a long poll, the various timing intervals in a request/response timing sequence (e.g., 'D,' 'RT,' and/or 'IT') can be sent to the proxy server 325 for use in polling the content source (e.g., application server/content host 110). The local proxy 275 can also identify to the proxy server 325 that a given application or request to be monitored is a long poll request (e.g., instructing the proxy server to set a 'long poll flag,' for example). In addition, the proxy server uses the various timing intervals to determine when to send keep-alive indications on behalf of mobile devices.

[00286] The local cache invalidator 244 of the caching policy manager 245 can invalidate cache elements in the local cache (e.g., cache 185 or 285) when new or changed data (e.g., updated response) is detected from the application server/content source for a given request. The cached response can be determined to be invalid for the outgoing request based on a notification received from the proxy server (e.g., proxy 325 or the host server 300). The source which provides responses to requests of the mobile client can be monitored to determine relevancy of the cached response stored in the cache of the mobile device 250 for the request. For example, the cache invalidator 244 can further remove/delete the cached response from the cache of the mobile device when the cached response is no longer valid for a given request or a given application.

[00287] In one embodiment, the cached response is removed from the cache after it is provided once again to an application which generated the outgoing request after determining that the cached response is no longer valid. The cached response can be provided again without waiting for the time interval or provided again after waiting for a time interval (e.g., the time interval determined to be specific to emulate the response delay in a long poll). In one embodiment, the time interval is the response delay 'D' or an average value of the response delay 'D' over two or more values.

[00288] The new or changed data can be, for example, detected by the proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 3A**). When a cache entry for a given request/poll has been invalidated, the use of the radio on the mobile device 250 can be enabled (e.g., by the local proxy 275 or the cache policy manager 245) to satisfy the subsequent polling requests.

[00289] One embodiment of the cache policy manager 245 includes a cache or connect selection engine 249 which can decide whether to use a locally cached entry to satisfy a poll/content request generated at the mobile device 250 by an application or widget. For example, the local proxy 275 or the cache policy manger 245 can intercept a polling request, made by an application (e.g., mobile application) on the mobile device, to contact the application server/content provider. The selection engine 249 can determine whether the content received for the intercepted request has been locally stored as cache elements for deciding whether the radio of the mobile device needs to be activated to satisfy the request made by the application (e.g., mobile application) and also determine whether the cached

response is still valid for the outgoing request prior to responding to the outgoing request using the cached response.

[00290] In one embodiment, the local proxy 275, in response to determining that relevant cached content exists and is still valid, can retrieve the cached elements from the local cache to provide a response to the application (e.g., mobile application) which made the polling request such that a radio of the mobile device is not activated to provide the response to the application (e.g., mobile application). In general, the local proxy 275 continues to provide the cached response each time the outgoing request is received until the updated response different from the cached response is detected.

[00291] When it is determined that the cached response is no longer valid, a new request for a given request is transmitted over the wireless network for an updated response. The request can be transmitted to the application server/content provider (e.g., server/host 110) or the proxy server on the host server (e.g., proxy 325 on the host 300) for a new and updated response. In one embodiment the cached response can be provided again as a response to the outgoing request if a new response is not received within the time interval, prior to removal of the cached response from the cache on the mobile device.

[00292] **FIG. 4C** depicts a block diagram illustrating another example of components in the application behavior detector 236 and the caching policy manager 245 in the local proxy 275 on the client-side of the distributed proxy system shown in the example of **FIG. 4A**. The illustrated application behavior detector 236 and the caching policy manager 245 can, for example, enable the local proxy 275 to detect cache defeat and perform caching of content addressed by identifiers intended to defeat cache.

[00293] In one embodiment, the caching policy manager 245 includes a cache defeat resolution engine 221, an identifier formalizer 211, a cache appropriateness decision engine 246, a poll schedule generator 247, an application protocol module 248, a cache or connect selection engine 249 having a cache query module 229, and/or a local cache invalidator 244. The cache defeat resolution engine 221 can further include a pattern extraction module 222 and/or a cache defeat parameter detector 223. The cache defeat parameter detector 223 can further include a random parameter detector 224 and/or a time/date parameter detector 226. One embodiment further includes an application cache policy repository 243 coupled to the decision engine 246.

[00294] In one embodiment, the application behavior detector 236 includes a pattern detector 237, a poll interval detector 238, an application profile generator 239, and/or a priority engine 241. The pattern detector 237 can further include a cache defeat parameter detector 223 having also, for example, a random parameter detector 233 and/or a time/date parameter detector 234. One embodiment further includes an application profile repository 242 coupled to the application profile generator 239. The application profile generator 239, and the priority engine 241 have been described in association with the description of the application behavior detector 236 in the example of **FIG.4A**.

[00295] The cache defeat resolution engine 221 can detect, identify, track, manage, and/or monitor content or content sources (e.g., servers or hosts) which employ identifiers and/or are addressed by identifiers (e.g., resource identifiers such as URLs and/or URIs) with one or more mechanisms that defeat cache or are intended to defeat cache. The cache defeat resolution engine 221 can, for example, detect from a given data request generated by an application or client that the identifier defeats or potentially defeats cache, where the data request otherwise addresses content or responses from a host or server (e.g., application server/content host 110 or 310) that is cacheable.

[00296] In one embodiment, the cache defeat resolution engine 221 detects or identifies cache defeat mechanisms used by content sources (e.g., application server/content host 110 or 310) using the identifier of a data request detected at the mobile device 250. The cache defeat resolution engine 221 can detect or identify a parameter in the identifier which can indicate that cache defeat mechanism is used. For example, a format, syntax, or pattern of the parameter can be used to identify cache defeat (e.g., a pattern, format, or syntax as determined or extracted by the pattern extraction module 222).

[00297] The pattern extraction module 222 can parse an identifier into multiple parameters or components and perform a matching algorithm on each parameter to identify any of which match one or more predetermined formats. For example, the results of the matching or the parsed out parameters from an identifier can be used (e.g., by the cache defeat parameter detector 223) to identify cache defeating parameters which can include one or more changing parameters.

[00298] The cache defeat parameter detector 223, in one embodiment can detect random parameters (e.g., by the random parameter detector 224) and/or time and/or date parameters

which are typically used for cache defeat. The cache defeat parameter detector 223 can detect random parameters and/or time/dates using commonly employed formats for these parameters and performing pattern matching algorithms and tests.

[00299] In addition to detecting patterns, formats, and/or syntaxes, the cache defeat parameter detector 223 further determines or confirms whether a given parameter is defeating cache and whether the addressed content can be cached by the distributed caching system. The cache defeat parameter detector 223 can detect this by analyzing responses received for the identifiers utilized by a given data request. In general, a changing parameter in the identifier is identified to indicate cache defeat when responses corresponding to multiple data requests are the same even when the multiple data requests uses identifiers with the changing parameter being different for each of the multiple data requests.

[00300] For example, at least two same responses may be required to identify the changing parameter as indicating cache defeat. In some instances, at least three same responses may be required. The requirement for the number of same responses needed to determine that a given parameter with a varying value between requests is cache defeating may be application specific, context dependent, and/or user dependent/user specified, or a combination of the above. Such a requirement may also be static or dynamically adjusted by the distributed cache system to meet certain performance thresholds and/or either explicit/implicit feedback regarding user experience (e.g., whether the user or application is receiving relevant/fresh content responsive to requests). More of the same responses may be required to confirm cache defeat, or for the system to treat a given parameter as intended for cache defeat if an application begins to malfunction due to response caching and/or if the user expresses dissatisfaction (explicit user feedback) or the system detects user frustration (implicit user cues).

[00301] The cache appropriateness decision engine 246 can detect, assess, or determine whether content from a content source (e.g., application server/content provider 110 in the example of **FIG. 1C**) with which a mobile device 250 interacts, has content that may be suitable for caching. In some instances, content from a given application server/content provider (e.g., the server/provider 110 of **FIG. 1C**) is determined to be suitable for caching based on a set of criteria (for example, criteria specifying time criticality of the content that is being requested from the content source). In one embodiment, the local proxy (e.g., the local proxy 175 or 275 of **FIG. 1C** and **FIG. 4A**) applies a selection criteria to store the content

from the host server which is requested by an application as cached elements in a local cache on the mobile device to satisfy subsequent requests made by the application.

[00302] The selection criteria can also include, by way of example, but not limitation, state of the mobile device indicating whether the mobile device is active or inactive, network conditions, and/or radio coverage statistics. The cache appropriateness decision engine 246 can be anyone or any combination of the criteria, and in any order, in identifying sources for which caching may be suitable.

[00303] Once application servers/content providers having identified or detected content that is potentially suitable for local caching on the mobile device 250, the cache policy manager 245 can proceed to cache the associated content received from the identified sources by storing content received from the content source as cache elements in a local cache (e.g., local cache 185 or 285 shown in the examples of **FIG. 1C** and **FIG. 4A**, respectively) on the mobile device 250. The content source can also be identified to a proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 5A**, respectively) remote from and in wireless communication with the mobile device 250 such that the proxy server can monitor the content source (e.g., application server/content provider 110) for new or changed data. Similarly, the local proxy (e.g., the local proxy 175 or 275 of **FIG. 1C** and **FIG. 4A**, respectively) can identify to the proxy server that content received from a specific application server/content provider is being stored as cached elements in the local cache.

[00304] In one embodiment, cache elements are stored in the local cache 285 as being associated with a normalized version of an identifier for an identifier employing one or more parameters intended to defeat cache. The identifier can be normalized by the identifier normalizer module 211 and the normalization process can include, by way of example, one or more of: converting the URI scheme and host to lower-case, capitalizing letters in percent-encoded escape sequences, removing a default port, and removing duplicate slashes.

[00305] In another embodiment, the identifier is normalized by removing the parameter for cache defeat and/or replacing the parameter with a static value which can be used to address or be associated with the cached response received responsive to a request utilizing the identifier by the normalizer 211 or the cache defeat parameter handler 212. For example, the cached elements stored in the local cache 285 (shown in **FIG. 4A**) can be identified using the normalized version of the identifier or a hash value of the normalized version of the identifier.

The hash value of an identifier or of the normalized identifier may be generated by the hash engine 213.

[00306] Once content has been locally cached, the cache policy manager 245 can, upon receiving future polling requests to contact the content server, retrieve the cached elements from the local cache to respond to the polling request made at the mobile device 250 such that a radio of the mobile device is not activated to service the polling request. Such servicing and fulfilling application (e.g., mobile application) requests locally via local cache entries allow for more efficient resource and mobile network traffic utilization and management since network bandwidth and other resources need not be used to request/receive poll responses which may have not changed from a response that has already been received at the mobile device 250.

[00307] One embodiment of the cache policy manager 245 includes a poll schedule generator 247 which can generate a polling schedule for one or more applications on the mobile device 250. The polling schedule can specify a polling interval that can be employed by the proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 5A**) in monitoring the content source for one or more applications. The polling schedule can be determined, for example, based on the interval between the polling requests directed to the content source from the mobile device. In one embodiment, the poll interval detector 238 of the application behavior detector can monitor polling requests directed to a content source from the mobile device 250 in order to determine an interval between the polling requests made from any or all application (e.g., mobile application).

[00308] In one embodiment, the cache policy manager 245 sends the polling schedule is sent to the proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 5A**) and can be used by the proxy server in monitoring the content source, for example, for changed or new content. The local cache invalidator 244 of the caching policy manager 245 can invalidate cache elements in the local cache (e.g., cache 185 or 285) when new or changed data is detected from the application server/content source for a given request. The new or changed data can be, for example, detected by the proxy server. When a cache entry for a given request/poll has been invalidated and/or removed (e.g., deleted from cache) after invalidation, the use of the radio on the mobile device 250 can be enabled (e.g., by the local proxy or the cache policy manager 245) to satisfy the subsequent polling requests, as further described with reference to the interaction diagram of **FIG. 6B**.

[00309] In another embodiment, the proxy server (e.g., proxy server 125 or 325 shown in the examples of **FIG. 1C** and **FIG. 5A**) uses a modified version of a resource identifier used in a data request to monitor a given content source (the application server/content host 110 of **FIG. 1B** and **FIG. 1C** to which the data request is addressed) for new or changed data. For example, in the instance where the content source or identifier is detected to employ cache defeat mechanisms, a modified (e.g., normalized) identifier can be used instead to poll the content source. The modified or normalized version of the identifier can be communicated to the proxy server by the caching policy manager 245, or more specifically the cache defeat parameter handler 212 of the identifier normalizer 211.

[00310] The modified identifier used by the proxy server to poll the content source on behalf of the mobile device/application (e.g., mobile application) may or may not be the same as the normalized identifier. For example, the normalized identifier may be the original identifier with the changing cache defeating parameter removed whereas the modified identifier uses a substitute parameter in place of the parameter that is used to defeat cache (e.g., the changing parameter replaced with a static value or other predetermined value known to the local proxy and/or proxy server). The modified parameter can be determined by the local proxy 275 and communicated to the proxy server. The modified parameter may also be generated by the proxy server (e.g., by the identifier modifier module 353 shown in the example of **FIG. 5C**).

[00311] One embodiment of the cache policy manager 245 includes a cache or connect selection engine 249 which can decide whether to use a locally cached entry to satisfy a poll/content request generated at the mobile device 250 by an application or widget. For example, the local proxy 275 or the cache policy manger 245 can intercept a polling request made by an application (e.g., mobile application) on the mobile device, to contact the application server/content provider. The selection engine 249 can determine whether the content received for the intercepted request has been locally stored as cache elements for deciding whether the a radio of the mobile device needs to be activated to satisfy the request made by the application (e.g., mobile application). In one embodiment, the local proxy 275, in response to determining that relevant cached content exists and is still valid, can retrieve the cached elements from the local cache to provide a response to the application (e.g., mobile application) which made the polling request such that a radio of the mobile device is not activated to provide the response to the application (e.g., mobile application).

[00312] In one embodiment, the cached elements stored in the local cache 285 (shown in **FIG. 4A**) can be identified using a normalized version of the identifier or a hash value of the normalized version of the identifier, for example, using the cache query module 229. Cached elements can be stored with normalized identifiers which have cache defeating parameters removed or otherwise replaced such that the relevant cached elements can be identified and retrieved in the future to satisfy other requests employing the same type of cache defeat. For example, when an identifier utilized in a subsequent request is determined to be utilizing the same cache defeating parameter, the normalized version of this identifier can be generated and used to identify a cached response stored in the mobile device cache to satisfy the data request. The hash value of an identifier or of the normalized identifier may be generated by the hash engine 213 of the identifier normalizer 211.

[00313] **FIG. 4D** depicts a block diagram illustrating examples of additional components in the local proxy 275 shown in the example of **FIG.4A** which is further capable of performing mobile traffic categorization and policy implementation based on application behavior and/or user activity.

[00314] In this embodiment of the local proxy 275, the user activity module 215 further includes one or more of, a user activity tracker 215a, a user activity prediction engine 215b, and/or a user expectation manager 215c. The application behavior detect 236 can further include a prioritization engine 241a, a time criticality detection engine 241b, an application state categorizer 241c, and/or an application traffic categorizer 241d. The local proxy 275 can further include a backlight detector 219 and/or a network configuration selection engine 251. The network configuration selection engine 251 can further include, one or more of, a wireless generation standard selector 251a, a data rate specifier 251b, an access channel selection engine 251c, and/or an access point selector.

[00315] In one embodiment, the application behavior detector 236 is able to detect, determined, identify, or infer, the activity state of an application on the mobile device 250 to which traffic has originated from or is directed to, for example, via the application state categorizer 241c and/or the traffic categorizer 241d. The activity state can be determined by whether the application is in a foreground or background state on the mobile device (via the application state categorizer 241c) since the traffic for a foreground application vs. a background application may be handled differently.

[00316] In one embodiment, the activity state can be determined, detected, identified, or inferred with a level of certainty of heuristics, based on the backlight status of the mobile device 250 (e.g., by the backlight detector 219) or other software agents or hardware sensors on the mobile device, including but not limited to, resistive sensors, capacitive sensors, ambient light sensors, motion sensors, touch sensors, etc. In general, if the backlight is on, the traffic can be treated as being or determined to be generated from an application that is active or in the foreground, or the traffic is interactive. In addition, if the backlight is on, the traffic can be treated as being or determined to be traffic from user interaction or user activity, or traffic containing data that the user is expecting within some time frame.

[00317] In one embodiment, the activity state is determined based on whether the traffic is interactive traffic or maintenance traffic. Interactive traffic can include transactions from responses and requests generated directly from user activity/interaction with an application and can include content or data that a user is waiting or expecting to receive. Maintenance traffic may be used to support the functionality of an application which is not directly detected by a user. Maintenance traffic can also include actions or transactions that may take place in response to a user action, but the user is not actively waiting for or expecting a response.

[00318] For example, a mail or message delete action at a mobile device 250 generates a request to delete the corresponding mail or message at the server, but the user typically is not waiting for a response. Thus, such a request may be categorized as maintenance traffic, or traffic having a lower priority (e.g., by the prioritization engine 241a) and/or is not time-critical (e.g., by the time criticality detection engine 214b).

[00319] Contrastingly, a mail 'read' or message 'read' request initiated by a user a the mobile device 250, can be categorized as 'interactive traffic' since the user generally is waiting to access content or data when they request to read a message or mail. Similarly, such a request can be categorized as having higher priority (e.g., by the prioritization engine 241a) and/or as being time critical/time sensitive (e.g., by the time criticality detection engine 241b).

[00320] The time criticality detection engine 241b can generally determine, identify, infer the time sensitivity of data contained in traffic sent from the mobile device 250 or to the mobile device from a host server (e.g., host 300) or application server (e.g., app server/content source 110). For example, time sensitive data can include, status updates, stock information updates, IM presence information, email messages or other messages, actions generated from

mobile gaming applications, webpage requests, location updates, etc. Data that is not time sensitive or time critical, by nature of the content or request, can include requests to delete messages, mark-as-read or edited actions, application-specific actions such as a add-friend or delete-friend request, certain types of messages, or other information which does not frequently changing by nature, etc. In some instances when the data is not time critical, the timing with which to allow the traffic to pass through is set based on when additional data needs to be sent from the mobile device 250. For example, traffic shaping engine 255 can align the traffic with one or more subsequent transactions to be sent together in a single power-on event of the mobile device radio (e.g., using the alignment module 256 and/or the batching module 257). The alignment module 256 can also align polling requests occurring close in time directed to the same host server, since these request are likely to be responded to with the same data.

[00321] In the alternate or in combination, the activity state can be determined from assessing, determining, evaluating, inferring, identifying user activity at the mobile device 250 (e.g., via the user activity module 215). For example, user activity can be directly detected and tracked using the user activity tracker 215a. The traffic resulting therefrom can then be categorized appropriately for subsequent processing to determine the policy for handling. Furthermore, user activity can be predicted or anticipated by the user activity prediction engine 215b. By predicting user activity or anticipating user activity, the traffic thus occurring after the prediction can be treated as resulting from user activity and categorized appropriately to determine the transmission policy.

[00322] In addition, the user activity module 215 can also manage user expectations (e.g., via the user expectation manager 215c and/or in conjunction with the activity tracker 215 and/or the prediction engine 215b) to ensure that traffic is categorized appropriately such that user expectations are generally met. For example, a user-initiated action should be analyzed (e.g., by the expectation manager 215) to determine or infer whether the user would be waiting for a response. If so, such traffic should be handled under a policy such that the user does not experience an unpleasant delay in receiving such a response or action.

[00323] In one embodiment, an advanced generation wireless standard network is selected for use in sending traffic between a mobile device and a host server in the wireless network based on the activity state of the application on the mobile device for which traffic is originated from or directed to. An advanced technology standards such as the 3G, 3.5G, 3G+, 4G, or LTE network can be selected for handling traffic generated as a result of user interaction, user

activity, or traffic containing data that the user is expecting or waiting for. Advanced generation wireless standard network can also be selected for to transmit data contained in traffic directed to the mobile device which responds to foreground activities.

[00324] In categorizing traffic and defining a transmission policy for mobile traffic, a network configuration can be selected for use (e.g., by the network configuration selection engine 251) on the mobile device 250 in sending traffic between the mobile device and a proxy server (325) and/or an application server (e.g., app server/host 110). The network configuration that is selected can be determined based on information gathered by the application behavior module 236 regarding application activity state (e.g., background or foreground traffic), application traffic category (e.g., interactive or maintenance traffic), any priorities of the data/content, time sensitivity/criticality.

[00325] The network configuration selection engine 2510 can select or specify one or more of, a generation standard (e.g., via wireless generation standard selector 251a), a data rate (e.g., via data rate specifier 251b), an access channel (e.g., access channel selection engine 251c), and/or an access point (e.g., via the access point selector 251d), in any combination.

[00326] For example, a more advanced generation (e.g., 3G, LTE, or 4G or later) can be selected or specified for traffic when the activity state is in interaction with a user or in a foreground on the mobile device. Contrastingly, an older generation standard (e.g., 2G, 2.5G, or 3G or older) can be specified for traffic when one or more of the following is detected, the application is not interacting with the user, the application is running in the background on the mobile device, or the data contained in the traffic is not time critical, or is otherwise determined to have lower priority.

[00327] Similarly, a network configuration with a slower data rate can be specified for traffic when one or more of the following is detected, the application is not interacting with the user, the application is running in the background on the mobile device, or the data contained in the traffic is not time critical. The access channel (e.g., Forward access channel or dedicated channel) can be specified.

[00328] **FIG. 5A** depicts a block diagram illustrating an example of server-side components in a distributed proxy and cache system residing on a host server 300 that manages traffic in a wireless network for resource conservation. The server-side proxy (or proxy server

325) can further categorize mobile traffic and/or implement delivery policies based on application behavior, content priority, user activity, and/or user expectations.

[00329] The host server 300 generally includes, for example, a network interface 308 and/or one or more repositories 312, 314, and 316. Note that server 300 may be any portable/mobile or non-portable device, server, cluster of computers and/or other types of processing units (e.g., any number of a machine shown in the example of **FIG. 27**) able to receive or transmit signals to satisfy data requests over a network including any wired or wireless networks (e.g., Wi-Fi, cellular, Bluetooth, etc.).

[00330] The network interface 308 can include networking module(s) or devices(s) that enable the server 300 to mediate data in a network with an entity that is external to the host server 300, through any known and/or convenient communications protocol supported by the host and the external entity. Specifically, the network interface 308 allows the server 300 to communicate with multiple devices including mobile phone devices 350 and/or one or more application servers/content providers 310.

[00331] The host server 300 can store information about connections (e.g., network characteristics, conditions, types of connections, etc.) with devices in the connection metadata repository 312. Additionally, any information about third party application or content providers can also be stored in the repository 312. The host server 300 can store information about devices (e.g., hardware capability, properties, device settings, device language, network capability, manufacturer, device model, OS, OS version, etc.) in the device information repository 314. Additionally, the host server 300 can store information about network providers and the various network service areas in the network service provider repository 316.

[00332] The communication enabled by network interface 308 allows for simultaneous connections (e.g., including cellular connections) with devices 350 and/or connections (e.g., including wired/wireless, HTTP, Internet connections, LAN, Wi-Fi, etc.) with content servers/providers 310 to manage the traffic between devices 350 and content providers 310, for optimizing network resource utilization and/or to conserver power (battery) consumption on the serviced devices 350. The host server 300 can communicate with mobile devices 350 serviced by different network service providers and/or in the same/different network service areas. The host server 300 can operate and is compatible with devices 350 with varying types or levels of mobile capabilities, including by way of example but not limitation, 1G, 2G, 2G

transitional (2.5G, 2.75G), 3G (IMT-2000), 3G transitional (3.5G, 3.75G, 3.9G), 4G (IMT-advanced), etc.

[00333] In general, the network interface 308 can include one or more of a network adaptor card, a wireless network interface card (e.g., SMS interface, Wi-Fi interface, interfaces for various generations of mobile communication standards including but not limited to 1G, 2G, 3G, 3.5G, 4G type networks such as LTE, WiMAX, etc.), Bluetooth, Wi-Fi, or any other network whether or not connected via a router, an access point, a wireless router, a switch, a multilayer switch, a protocol converter, a gateway, a bridge, a bridge router, a hub, a digital media receiver, and/or a repeater.

[00334] The host server 300 can further include server-side components of the distributed proxy and cache system which can include a proxy server 325 and a server cache 335. In one embodiment, the proxy server 325 can include an HTTP access engine 345, a caching policy manager 355, a proxy controller 365, a traffic shaping engine 375, a new data detector 347 and/or a connection manager 395.

[00335] The HTTP access engine 345 may further include a heartbeat manager 398; the proxy controller 365 may further include a data invalidator module 368; the traffic shaping engine 375 may further include a control protocol 376 and a batching module 377. Additional or less components/modules/engines can be included in the proxy server 325 and each illustrated component.

[00336] As used herein, a "module," a "manager," a "handler," a "detector," an "interface," a "controller," a "normalizer," a "generator," an "invalidator," or an "engine" includes a general purpose, dedicated or shared processor and, typically, firmware or software modules that are executed by the processor. Depending upon implementation-specific or other considerations, the module, manager, handler, detector, interface, controller, normalizer, generator, invalidator, or engine can be centralized or its functionality distributed. The module, manager, handler, detector, interface, controller, normalizer, generator, invalidator, or engine can include general or special purpose hardware, firmware, or software embodied in a computer-readable (storage) medium for execution by the processor. As used herein, a computer-readable medium or computer-readable storage medium is intended to include all mediums that are statutory (e.g., in the United States, under 35 U.S.C. 101), and to specifically exclude all mediums that are non-statutory in nature to the extent that the exclusion is

necessary for a claim that includes the computer-readable (storage) medium to be valid. Known statutory computer-readable mediums include hardware (e.g., registers, random access memory (RAM), non-volatile (NV) storage, to name a few), but may or may not be limited to hardware.

[00337] In the example of a device (e.g., mobile device 350) making an application or content request to an application server or content provider 310, the request may be intercepted and routed to the proxy server 325 which is coupled to the device 350 and the application server/content provider 310. Specifically, the proxy server is able to communicate with the local proxy (e.g., proxy 175 and 275 of the examples of **FIG. 1** and **FIG. 2** respectively) of the mobile device 350, the local proxy forwards the data request to the proxy server 325 in some instances for further processing and, if needed, for transmission to the application server/content server 310 for a response to the data request.

[00338] In such a configuration, the host 300, or the proxy server 325 in the host server 300 can utilize intelligent information provided by the local proxy in adjusting its communication with the device in such a manner that optimizes use of network and device resources. For example, the proxy server 325 can identify characteristics of user activity on the device 350 to modify its communication frequency. The characteristics of user activity can be determined by, for example, the activity/behavior awareness module 366 in the proxy controller 365 via information collected by the local proxy on the device 350.

[00339] In one embodiment, communication frequency can be controlled by the connection manager 395 of the proxy server 325, for example, to adjust push frequency of content or updates to the device 350. For instance, push frequency can be decreased by the connection manager 395 when characteristics of the user activity indicate that the user is inactive. In one embodiment, when the characteristics of the user activity indicate that the user is subsequently active after a period of inactivity, the connection manager 395 can adjust the communication frequency with the device 350 to send data that was buffered as a result of decreased communication frequency to the device 350.

[00340] In addition, the proxy server 325 includes priority awareness of various requests, transactions, sessions, applications, and/or specific events. Such awareness can be determined by the local proxy on the device 350 and provided to the proxy server 325. The priority awareness module 367 of the proxy server 325 can generally assess the priority (e.g., including

time-criticality, time-sensitivity, etc.) of various events or applications; additionally, the priority awareness module 367 can track priorities determined by local proxies of devices 350.

[00341] In one embodiment, through priority awareness, the connection manager 395 can further modify communication frequency (e.g., use or radio as controlled by the radio controller 396) of the server 300 with the devices 350. For example, the server 300 can notify the device 350, thus requesting use of the radio if it is not already in use when data or updates of an importance/priority level which meets a criteria becomes available to be sent.

[00342] In one embodiment, the proxy server 325 can detect multiple occurrences of events (e.g., transactions, content, data received from server/provider 310) and allow the events to accumulate for batch transfer to device 350. Batch transfer can be cumulated and transfer of events can be delayed based on priority awareness and/or user activity/application behavior awareness as tracked by modules 367 and/or 366. For example, batch transfer of multiple events (of a lower priority) to the device 350 can be initiated by the batching module 377 when an event of a higher priority (meeting a threshold or criteria) is detected at the server 300. In addition, batch transfer from the server 300 can be triggered when the server receives data from the device 350, indicating that the device radio is already in use and is thus on. In one embodiment, the proxy server 325 can order the each messages/packets in a batch for transmission based on event/transaction priority such that higher priority content can be sent first in case connection is lost or the battery dies, etc.

[00343] In one embodiment, the server 300 caches data (e.g., as managed by the caching policy manager 355) such that communication frequency over a network (e.g., cellular network) with the device 350 can be modified (e.g., decreased). The data can be cached, for example, in the server cache 335 for subsequent retrieval or batch sending to the device 350 to potentially decrease the need to turn on the device 350 radio. The server cache 335 can be partially or wholly internal to the host server 300, although in the example of **FIG. 5A** it is shown as being external to the host 300. In some instances, the server cache 335 may be the same as and/or integrated in part or in whole with another cache managed by another entity (e.g., the optional caching proxy server 199 shown in the example of **FIG. 1C**), such as being managed by an application server/content provider 310, a network service provider, or another third party.

[00344] In one embodiment, content caching is performed locally on the device 350 with the assistance of host server 300. For example, proxy server 325 in the host server 300 can query the application server/provider 310 with requests and monitor changes in responses. When changed or new responses are detected (e.g., by the new data detector 347), the proxy server 325 can notify the mobile device 350 such that the local proxy on the device 350 can make the decision to invalidate (e.g., indicated as out-dated) the relevant cache entries stored as any responses in its local cache. Alternatively, the data invalidator module 368 can automatically instruct the local proxy of the device 350 to invalidate certain cached data, based on received responses from the application server/provider 310. The cached data is marked as invalid, and can get replaced or deleted when new content is received from the content server 310.

[00345] Note that data change can be detected by the detector 347 in one or more ways. For example, the server/provider 310 can notify the host server 300 upon a change. The change can also be detected at the host server 300 in response to a direct poll of the source server/provider 310. In some instances, the proxy server 325 can in addition, pre-load the local cache on the device 350 with the new/updated data. This can be performed when the host server 300 detects that the radio on the mobile device is already in use, or when the server 300 has additional content/data to be sent to the device 350.

[00346] One or more the above mechanisms can be implemented simultaneously or adjusted/configured based on application (e.g., different policies for different servers/providers 310). In some instances, the source provider/server 310 may notify the host 300 for certain types of events (e.g., events meeting a priority threshold level). In addition, the provider/server 310 may be configured to notify the host 300 at specific time intervals, regardless of event priority.

[00347] In one embodiment, the proxy server 325 of the host 300 can monitor/track responses received for the data request from the content source for changed results prior to returning the result to the mobile device, such monitoring may be suitable when data request to the content source has yielded same results to be returned to the mobile device, thus preventing network/power consumption from being used when no new changes are made to a particular requested. The local proxy of the device 350 can instruct the proxy server 325 to perform such monitoring or the proxy server 325 can automatically initiate such a process upon receiving a

certain number of the same responses (e.g., or a number of the same responses in a period of time) for a particular request.

[00348] In one embodiment, the server 300, through the activity/behavior awareness module 366, is able to identify or detect user activity at a device that is separate from the mobile device 350. For example, the module 366 may detect that a user's message inbox (e.g., email or types of inbox) is being accessed. This can indicate that the user is interacting with his/her application using a device other than the mobile device 350 and may not need frequent updates, if at all.

[00349] The server 300, in this instance, can thus decrease the frequency with which new or updated content is sent to the mobile device 350, or eliminate all communication for as long as the user is detected to be using another device for access. Such frequency decrease may be application specific (e.g., for the application with which the user is interacting with on another device), or it may be a general frequency decrease (E.g., since the user is detected to be interacting with one server or one application via another device, he/she could also use it to access other services.) to the mobile device 350.

[00350] In one embodiment, the host server 300 is able to poll content sources 310 on behalf of devices 350 to conserve power or battery consumption on devices 350. For example, certain applications on the mobile device 350 can poll its respective server 310 in a predictable recurring fashion. Such recurrence or other types of application behaviors can be tracked by the activity/behavior module 366 in the proxy controller 365. The host server 300 can thus poll content sources 310 for applications on the mobile device 350 that would otherwise be performed by the device 350 through a wireless (e.g., including cellular connectivity). The host server can poll the sources 310 for new or changed data by way of the HTTP access engine 345 to establish HTTP connection or by way of radio controller 396 to connect to the source 310 over the cellular network. When new or changed data is detected, the new data detector 347 can notify the device 350 that such data is available and/or provide the new/changed data to the device 350.

[00351] In one embodiment, the connection manager 395 determines that the mobile device 350 is unavailable (e.g., the radio is turned off) and utilizes SMS to transmit content to the device 350, for instance, via the SMSC shown in the example of **FIG. 1C**. SMS is used to transmit invalidation messages, batches of invalidation messages, or even content in the case

where the content is small enough to fit into just a few (usually one or two) SMS messages. This avoids the need to access the radio channel to send overhead information. The host server 300 can use SMS for certain transactions or responses having a priority level above a threshold or otherwise meeting a criteria. The server 300 can also utilize SMS as an out-of-band trigger to maintain or wake-up an IP connection as an alternative to maintaining an always-on IP connection.

[00352] In one embodiment, the connection manager 395 in the proxy server 325 (e.g., the heartbeat manager 398) can generate and/or transmit heartbeat messages on behalf of connected devices 350 to maintain a backend connection with a provider 310 for applications running on devices 350.

[00353] For example, in the distributed proxy system, local cache on the device 350 can prevent any or all heartbeat messages needed to maintain TCP/IP connections required for applications from being sent over the cellular, or other, network and instead rely on the proxy server 325 on the host server 300 to generate and/or send the heartbeat messages to maintain a connection with the backend (e.g., application server/provider 110 in the example of **FIG. 1B**). The proxy server can generate the keep-alive (heartbeat) messages independent of the operations of the local proxy on the mobile device.

[00354] The repositories 312, 314, and/or 316 can additionally store software, descriptive data, images, system information, drivers, and/or any other data item utilized by other components of the host server 300 and/or any other servers for operation. The repositories may be managed by a database management system (DBMS), for example, which may be but is not limited to Oracle, DB2, Microsoft Access, Microsoft SQL Server, PostgreSQL, MySQL, FileMaker, etc.

[00355] The repositories can be implemented via object-oriented technology and/or via text files and can be managed by a distributed database management system, an object-oriented database management system (OODBMS) (e.g., ConceptBase, FastDB Main Memory Database Management System, JDOInstruments, ObjectDB, etc.), an object-relational database management system (ORDBMS) (e.g., Informix, OpenLink Virtuoso, VMDS, etc.), a file system, and/or any other convenient or known database management package.

[00356] **FIG. 5B** depicts a block diagram illustrating a further example of components in the caching policy manager 355 in the cache system shown in the example of **FIG. 5A** which

is capable of caching and adapting caching strategies for application (e.g., mobile application) behavior and/or network conditions.

[00357] The caching policy manager 355, in one embodiment, can further include a metadata generator 303, a cache look-up engine 305, an application protocol module 356, a content source monitoring engine 357 having a poll schedule manager 358, a response analyzer 361, and/or an updated or new content detector 359. In one embodiment, the poll schedule manager 358 further includes a host timing simulator 358a, a long poll request detector/manager 358b, a schedule update engine 358c, and/or a time adjustment engine 358d. The metadata generator 303 and/or the cache look-up engine 305 can be coupled to the cache 335 (or, server cache) for modification or addition to cache entries or querying thereof.

[00358] In one embodiment, the proxy server (e.g., the proxy server 125 or 325 of the examples of **FIG. 1C** and **FIG. 5A**) can monitor a content source for new or changed data via the monitoring engine 357. The proxy server, as shown, is an entity external to the mobile device 250 of **FIG. 4A-B**. The content source (e.g., application server/content provider 110 of **FIG. 1C**) can be one that has been identified to the proxy server (e.g., by the local proxy) as having content that is being locally cached on a mobile device (e.g., mobile device 150 or 250). The content source can be monitored, for example, by the monitoring engine 357 at a frequency that is based on polling frequency of the content source at the mobile device. The poll schedule can be generated, for example, by the local proxy and sent to the proxy server. The poll frequency can be tracked and/or managed by the poll schedule manager 358.

[00359] For example, the proxy server can poll the host (e.g., content provider/application server) on behalf of the mobile device and simulate the polling behavior of the client to the host via the host timing simulator 358a. The polling behavior can be simulated to include characteristics of a long poll request-response sequences experienced in a persistent connection with the host (e.g., by the long poll request detector/manager 358b). Note that once a polling interval/behavior is set, the local proxy 275 on the device-side and/or the proxy server 325 on the server-side can verify whether application and application server/content host behavior match or can be represented by this predicted pattern. In general, the local proxy and/or the proxy server can detect deviations and, when appropriate, re-evaluate and compute, determine, or estimate another polling interval.

[00360] In one embodiment, the caching policy manager 355 on the server-side of the distribute proxy can, in conjunction with or independent of the proxy server 275 on the mobile device, identify or detect long poll requests. For example, the caching policy manager 355 can determine a threshold value to be used in comparison with a response delay interval time (interval time 'D') in a request-response sequence for an application request to identify or detect long poll requests, possible long poll requests (e.g., requests for a persistent connection with a host with which the client communicates including, but not limited to, a long-held HTTP request, a persistent connection enabling COMET style push, request for HTTP streaming, etc.), or other requests which can otherwise be treated as a long poll request.

[00361] For example, the threshold value can be determined by the proxy 325 using response delay interval times for requests generated by clients/applications across mobile devices which may be serviced by multiple different cellular or wireless networks. Since the proxy 325 resides on host 300 is able to communicate with multiple mobile devices via multiple networks, the caching policy manager 355 has access to application/client information at a global level which can be used in setting threshold values to categorize and detect long polls.

[00362] By tracking response delay interval times across applications across devices over different or same networks, the caching policy manager 355 can set one or more threshold values to be used in comparison with response delay interval times for long poll detection. Threshold values set by the proxy server 325 can be static or dynamic, and can be associated with conditions and/or a time-to-live (an expiration time/date in relative or absolute terms).

[00363] In addition, the caching policy manager 355 of the proxy 325 can further determine the threshold value, in whole or in part, based on network delays of a given wireless network, networks serviced by a given carrier (service provider), or multiple wireless networks. The proxy 325 can also determine the threshold value for identification of long poll requests based on delays of one or more application server/content provider (e.g., 110) to which application (e.g., mobile application) or mobile client requests are directed.

[00364] The proxy server can detect new or changed data at a monitored content source and transmits a message to the mobile device notifying it of such a change such that the mobile device (or the local proxy on the mobile device) can take appropriate action (e.g., to invalidate the cache elements in the local cache). In some instances, the proxy server (e.g., the caching

policy manager 355) upon detecting new or changed data can also store the new or changed data in its cache (e.g., the server cache 135 or 335 of the examples of **FIG. 1C** and **FIG. 5A**, respectively). The new/updated data stored in the server cache 335 can be used in some instances to satisfy content requests at the mobile device; for example, it can be used after the proxy server has notified the mobile device of the new/changed content and that the locally cached content has been invalidated.

[00365] The metadata generator 303, similar to the metadata generator 203 shown in the example of **FIG. 4B**, can generate metadata for responses cached for requests at the mobile device 250. The metadata generator 303 can generate metadata for cache entries stored in the server cache 335. Similarly, the cache look-up engine 305 can include the same or similar functions are those described for the cache look-up engine 205 shown in the example of **FIG. 4B**.

[00366] The response analyzer 361 can perform any or all of the functionalities related to analyzing responses received for requests generated at the mobile device 250 in the same or similar fashion to the response analyzer 246d of the local proxy shown in the example of **FIG. 4B**. Since the proxy server 325 is able to receive responses from the application server/content source 310 directed to the mobile device 250, the proxy server 325 (e.g., the response analyzer 361) can perform similar response analysis steps to determine cacheability, as described for the response analyzer of the local proxy. The responses can be analyzed in addition to or in lieu of the analysis that can be performed at the local proxy 275 on the mobile device 250.

[00367] Furthermore, the schedule update engine 358c can update the polling interval of a given application server/content host based on application request interval changes of the application at the mobile device 250 as described for the schedule update engine in the local proxy 275. The time adjustment engine 358d can set an initial time at which polls of the application server/content host is to begin to prevent the serving of out of date content once again before serving fresh content as described for the schedule update engine in the local proxy 275. Both the schedule updating and the time adjustment algorithms can be performed in conjunction with or in lieu of the similar processes performed at the local proxy 275 on the mobile device 250.

[00368] **FIG. 5C** depicts a block diagram illustrating another example of components in the caching policy manager 355 in the proxy server 375 on the server-side of the distributed

proxy system shown in the example of **FIG. 5A** which is capable of managing and detecting cache defeating mechanisms and monitoring content sources.

[00369] The caching policy manager 355, in one embodiment, can further include a cache defeating source manager 352, a content source monitoring engine 357 having a poll schedule manager 358, and/or an updated or new content detector 359. The cache defeating source manager 352 can further include an identifier modifier module 353 and/or an identifier pattern tracking module 354.

[00370] In one embodiment, the proxy server (e.g., the proxy server 125 or 325 of the examples of **FIG. 1C** and **FIG. 5A**) can monitor a content source for new or changed data via the monitoring engine 357. The content source (e.g., application server/content provider 110 of **FIG. 1C** or 310 of **FIG. 5A**) can be one that has been identified to the proxy server (e.g., by the local proxy) as having content that is being locally cached on a mobile device (e.g., mobile device 150 or 250). The content source 310 can be monitored, for example, by the monitoring engine 357 at a frequency that is based on polling frequency of the content source at the mobile device. The poll schedule can be generated, for example, by the local proxy and sent to the proxy server 325. The poll frequency can be tracked and/or managed by the poll schedule manager 358.

[00371] In one embodiment, the proxy server 325 uses a normalized identifier or modified identifier in polling the content source 310 to detect new or changed data (responses). The normalized identifier or modified identifier can also be used by the proxy server 325 in storing responses on the server cache 335. In general, the normalized or modified identifiers can be used when cache defeat mechanisms are employed for cacheable content. Cache defeat mechanisms can be in the form of a changing parameter in an identifier such as a URI or URL and can include a changing time/data parameter, a randomly varying parameter, or other types parameters.

[00372] The normalized identifier or modified identifier removes or otherwise replaces the changing parameter for association with subsequent requests and identification of associated responses and can also be used to poll the content source. In one embodiment, the modified identifier is generated by the cache defeating source manager 352 (e.g., the identifier modifier module 353) of the caching policy manager 355 on the proxy server 325 (server-side component of the distributed proxy system). The modified identifier can utilize a substitute

parameter (which is generally static over a period of time) in place of the changing parameter that is used to defeat cache.

[00373] The cache defeating source manager 352 optionally includes the identifier pattern tracking module 354 to track, store, and monitor the various modifications of an identifier or identifiers that address content for one or more content sources (e.g., application server/content host 110 or 310) to continuously verify that the modified identifiers and/or normalized identifiers used by the proxy server 325 to poll the content sources work as predicted or intended (e.g., receive the same responses or responses that are otherwise still relevant compared to the original, unmodified identifier).

[00374] In the event that the pattern tracking module 354 detects a modification or normalization of an identifier that causes erratic or unpredictable behavior (e.g., unexpected responses to be sent) on the content source, the tracking module 354 can log the modification and instruct the cache defeating source manager 352 to generate another modification/normalization, or notify the local proxy (e.g., local proxy 275) to generate another modification/normalization for use in polling the content source. In the alternative or in parallel, the requests from the given mobile application/client on the mobile device (e.g., mobile device 250) can temporarily be sent across the network to the content source for direct responses to be provided to the mobile device and/or until a modification of an identifier which works can be generated.

[00375] In one embodiment, responses are stored as server cache elements in the server cache when new or changed data is detected for a response that is already stored on a local cache (e.g., cache 155) of the mobile device (e.g., mobile device 250). Therefore, the mobile device or local proxy 275 can connect to the proxy server 325 to retrieve the new or changed data for a response to a request which was previously cached locally in the local cache 155 (now invalid, out-dated, or otherwise determined to be irrelevant).

[00376] The proxy server 325 can detect new or changed data at a monitored application server/content host 310 and transmits a message to the mobile device notifying it of such a change such that the mobile device (or the local proxy on the mobile device) can take appropriate action (e.g., to invalidate the cache elements in the local cache). In some instances, the proxy server (e.g., the caching policy manager 355), upon detecting new or changed data, can also store the new or changed data in its cache (e.g., the server cache 135 or 335 of the

examples of **FIG. 1C** and **FIG. 5A**, respectively). The updated/new data stored in the server cache can be used, in some instances, to satisfy content requests at the mobile device; for example, it can be used after the proxy server has notified the mobile device of the new/changed content and that the locally cached content has been invalidated.

[00377] **FIG. 5D** depicts a block diagram illustrating examples of additional components in proxy server 325 shown in the example of **FIG. 5A** which is further capable of performing mobile traffic categorization and policy implementation based on application behavior and/or traffic priority.

[00378] In one embodiment of the proxy server 325, the traffic shaping engine 375 is further coupled to a traffic analyzer 336 for categorizing mobile traffic for policy definition and implementation for mobile traffic and transactions directed to one or more mobile devices (e.g., mobile device 250 of **FIG. 4A-4D**) or to an application server/content host (e.g., 110 of **FIG. 1B-1C**). In general, the proxy server 325 is remote from the mobile devices and remote from the host server, as shown in the examples of **FIG. 1B-1C**. The proxy server 325 or the host server 300 can monitor the traffic for multiple mobile devices and is capable of categorizing traffic and devising traffic policies for different mobile devices.

[00379] In addition, the proxy server 325 or host server 300 can operate with multiple carriers or network operators and can implement carrier-specific policies relating to categorization of traffic and implementation of traffic policies for the various categories. For example, the traffic analyzer 336 of the proxy server 325 or host server 300 can include one or more of, a prioritization engine 341a, a time criticality detection engine 341b, an application state categorizer 341c, and/or an application traffic categorizer 341d.

[00380] Each of these engines or modules can track different criterion for what is considered priority, time critical, background/foreground, or interactive/maintenance based on different wireless carriers. Different criterion may also exist for different mobile device types (e.g., device model, manufacturer, operating system, etc.). In some instances, the user of the mobile devices can adjust the settings or criterion regarding traffic category and the proxy server 325 is able to track and implement these user adjusted/configured settings.

[00381] In one embodiment, the traffic analyzer 336 is able to detect, determined, identify, or infer, the activity state of an application on one or more mobile devices (e.g., mobile device 150 or 250) which traffic has originated from or is directed to, for example, via the application

state categorizer 341c and/or the traffic categorizer 341d. The activity state can be determined based on whether the application is in a foreground or background state on one or more of the mobile devices (via the application state categorizer 341c) since the traffic for a foreground application vs. a background application may be handled differently to optimize network use.

[00382] In the alternate or in combination, the activity state of an application can be determined by the wirelessly connected mobile devices (e.g., via the application behavior detectors in the local proxies) and communicated to the proxy server 325. For example, the activity state can be determined, detected, identified, or inferred with a level of certainty of heuristics, based on the backlight status at mobile devices (e.g., by a backlight detector) or other software agents or hardware sensors on the mobile device, including but not limited to, resistive sensors, capacitive sensors, ambient light sensors, motion sensors, touch sensors, etc. In general, if the backlight is on, the traffic can be treated as being or determined to be generated from an application that is active or in the foreground, or the traffic is interactive. In addition, if the backlight is on, the traffic can be treated as being or determined to be traffic from user interaction or user activity, or traffic containing data that the user is expecting within some time frame.

[00383] The activity state can be determined from assessing, determining, evaluating, inferring, identifying user activity at the mobile device 250 (e.g., via the user activity module 215) and communicated to the proxy server 325. In one embodiment, the activity state is determined based on whether the traffic is interactive traffic or maintenance traffic. Interactive traffic can include transactions from responses and requests generated directly from user activity/interaction with an application and can include content or data that a user is waiting or expecting to receive. Maintenance traffic may be used to support the functionality of an application which is not directly detected by a user. Maintenance traffic can also include actions or transactions that may take place in response to a user action, but the user is not actively waiting for or expecting a response.

[00384] The time criticality detection engine 341b can generally determine, identify, infer the time sensitivity of data contained in traffic sent from the mobile device 250 or to the mobile device from the host server 300 or proxy server 325, or the application server (e.g., app server/content source 110). For example, time sensitive data can include, status updates, stock information updates, IM presence information, email messages or other messages, actions generated from mobile gaming applications, webpage requests, location updates, etc.

[00385] Data that is not time sensitive or time critical, by nature of the content or request, can include requests to delete messages, mark-as-read or edited actions, application-specific actions such as a add-friend or delete-friend request, certain types of messages, or other information which does not frequently changing by nature, etc. In some instances when the data is not time critical, the timing with which to allow the traffic to be sent to a mobile device is based on when there is additional data that needs to the sent to the same mobile device. For example, traffic shaping engine 375 can align the traffic with one or more subsequent transactions to be sent together in a single power-on event of the mobile device radio (e.g., using the alignment module 378 and/or the batching module 377). The alignment module 378 can also align polling requests occurring close in time directed to the same host server, since these request are likely to be responded to with the same data.

[00386] In general, whether new or changed data is sent from a host server to a mobile device can be determined based on whether an application on the mobile device to which the new or changed data is relevant, is running in a foreground (e.g., by the application state categorizer 341c), or the priority or time criticality of the new or changed data. The proxy server 325 can send the new or changed data to the mobile device if the application is in the foreground on the mobile device, or if the application is in the foreground and in an active state interacting with a user on the mobile device, and/or whether a user is waiting for a response that would be provided in the new or changed data. The proxy server 325 (or traffic shaping engine 375) can send the new or changed data that is of a high priority or is time critical.

[00387] Similarly, the proxy server 325 (or the traffic shaping engine 375) can suppressing the sending of the new or changed data if the application is in the background on the mobile device. The proxy server 325 can also suppress the sending of the new or changed data if the user is not waiting for the response provided in the new or changed data; wherein the suppressing is performed by a proxy server coupled to the host server and able to wirelessly connect to the mobile device.

[00388] In general, if data, including new or change data is of a low priority or is not time critical, the proxy server can waiting to transfer the data until after a time period, or until there is additional data to be sent (e.g., via the alignment module 378 and/or the batching module 377).

[00389] **FIG. 6A** depicts another flow diagram illustrating an example process for distributed content caching between a mobile device and a proxy server and the distributed management of content caching.

[00390] As shown in the distributed system interaction diagram in the example of **FIG. 4A-4D**, the disclosed technology is a distributed caching model with various aspects of caching tasks split between the client-side/mobile device side (e.g., mobile device 450 in the example of **FIG. 4A-4D**) and the server side (e.g., server side 470 including the host server 485 and/or the optional caching proxy 475).

[00391] In general the device-side responsibilities can include deciding whether a response to a particular request can be and/or should be cached. The device-side of the proxy can make this decision based on information (e.g., timing characteristics, detected pattern, detected pattern with heuristics, indication of predictability or repeatability) collected from/during both request and response and cache it (e.g., storing it in a local cache on the mobile device). The device side can also notify the server-side in the distributed cache system of the local cache event and notify it monitor the content source (e.g., application server/content provider 110 of **FIG. 1B-C**).

[00392] The device side can further instruct the server side of the distributed proxy to periodically validate the cache response (e.g., by way of polling, or sending polling requests to the content source). The device side can further decide whether a response to a particular cache request should be returned from the local cache (e.g., whether a cache hit is detected). The decision can be made by the device side (e.g., the local proxy on the device) using information collected from/during request and/or responses received from the content source.

[00393] In general, the server-side responsibilities can include validating cached responses for relevancy (e.g., determine whether a cached response is still valid or relevant to its associated request). The server-side can send the mobile device an invalidation request to notify the device side when a cached response is detected to be no longer valid or no longer relevant (e.g., the server invalidates a given content source). The device side then can remove the response from the local cache.

[00394] The diagram of **FIG. 6A** illustrates caching logic processes performed for each detected or intercepted request (e.g., HTTP request) detected at a mobile device (e.g., client-side of the distributed proxy). In step 602, the client-side of the proxy (e.g., local proxy 275

shown in **FIG. 2A-B** or mobile device 450 of **FIG. 4A-4D**) receives a request (from an application (e.g., mobile application) or mobile client). In step 604, URL is normalized and in step 606 the client-side checks to determine if the request is cacheable. If the request is determined to be not cacheable in step 612, the request is sent to the source (application server/content provider) in step 608 and the response is received 610 and delivered to the requesting application 622, similar to a request-response sequence without interception by the client side proxy.

[00395] If the request is determined to be cacheable, in step 612, the client-side looks up the cache to determine whether a cache entry exists for the current request. If so, in step 624, the client-side can determine whether the entry is valid and if so, the client side can check the request to see if includes a validator (e.g., a modified header or an entity tag) in step 615. For example, the concept of validation is eluded to in section 13.3 of RFC 2616 which describes in possible types of headers (e.g., eTAG, Modified_Since, must_revlaidate, pragma no_cache) and forms a validating response 632 if so to be delivered to the requesting application in step 622. If the request does not include a validator as determined by step 615, a response is formed from the local cache in step 630 and delivered to the requesting application in step 622. This validation step can be used for content that would otherwise normally be considered un-cacheable.

[00396] If, instead, in step 624, the cache entry is found but determined to be no longer valid or invalid, the client side of the proxy sends the request 616 to the content source (application server/content host) and receives a response directly fro the source in step 618. Similarly, if in step 612, a cache entry was not found during the look up, the request is also sent in step 616. Once the response is received, the client side checks the response to determine if it is cacheable in step 626. If so, the response is cached in step 620. The client then sends another poll in step 614 and then delivers the response to the requesting application in step 622.

[00397] **FIG. 6B** depicts a diagram showing how data requests from a mobile device 450 to an application server/content provider 495 in a wireless network can be coordinated by a distributed proxy system 460 in a manner such that network and battery resources are conserved through using content caching and monitoring performed by the distributed proxy system 460.

[00398] In satisfying application or client requests on a mobile device 450 without the distributed proxy system 460, the mobile device 450, or the software widget executing on the device 450, performs a data request 452 (e.g., an HTTP GET, POST, or other request) directly to the application server 495 and receives a response 404 directly from the server/provider 495. If the data has been updated, the widget 455 on the mobile device 450 can refreshes itself to reflect the update and waits for small period of time and initiates another data request to the server/provider 495.

[00399] In one embodiment, the requesting client or software widget 455 on the device 450 can utilize the distributed proxy system 460 in handling the data request made to server/provider 495. In general, the distributed proxy system 460 can include a local proxy 465 (which is typically considered a client-side component of the system 460 and can reside on the mobile device 450), a caching proxy 475 (considered a server-side component 470 of the system 460 and can reside on the host server 485 or be wholly or partially external to the host server 485), and a host server 485. The local proxy 465 can be connected to the caching proxy 475 and host server 485 via any network or combination of networks.

[00400] When the distributed proxy system 460 is used for data/application requests, the widget 455 can perform the data request 456 via the local proxy 465. The local proxy 465, can intercept the requests made by device applications, and can identify the connection type of the request (e.g., an HTTP get request or other types of requests). The local proxy 465 can then query the local cache for any previous information about the request (e.g., to determine whether a locally stored response is available and/or still valid). If a locally stored response is not available or if there is an invalid response stored, the local proxy 465 can update or store information about the request, the time it was made, and any additional data, in the local cache. The information can be updated for use in potentially satisfying subsequent requests.

[00401] The local proxy 465 can then send the request to the host server 485 and the host server 485 can perform the request 456 and returns the results in response 458. The local proxy 465 can store the result and, in addition, information about the result and returns the result to the requesting widget 455.

[00402] In one embodiment, if the same request has occurred multiple times (within a certain time period) and it has often yielded same results, the local proxy 465 can notify 460

the server 485 that the request should be monitored (e.g., steps 462 and 464) for result changes prior to returning a result to the local proxy 465 or requesting widget 455.

[00403] In one embodiment, if a request is marked for monitoring, the local proxy 465 can now store the results into the local cache. Now, when the data request 466, for which a locally response is available, is made by the widget 455 and intercepted at the local proxy 465, the local proxy 465 can return the response 468 from the local cache without needing to establish a connection communication over the wireless network.

[00404] In addition, the server proxy performs the requests marked for monitoring 470 to determine whether the response 472 for the given request has changed. In general, the host server 485 can perform this monitoring independently of the widget 455 or local proxy 465 operations. Whenever an unexpected response 472 is received for a request, the server 485 can notify the local proxy 465 that the response has changed (e.g., the invalidate notification in step 474) and that the locally stored response on the client should be erased or replaced with a new response.

[00405] In this case, a subsequent data request 476 by the widget 455 from the device 450 results in the data being returned from host server 485 (e.g., via the caching proxy 475), and in step 478, the request is satisfied from the caching proxy 475. Thus, through utilizing the distributed proxy system 460, the wireless (cellular) network is intelligently used when the content/data for the widget or software application 455 on the mobile device 450 has actually changed. As such, the traffic needed to check for the changes to application data is not performed over the wireless (cellular) network. This reduces the amount of generated network traffic and shortens the total time and the number of times the radio module is powered up on the mobile device 450, thus reducing battery consumption and, in addition, frees up network bandwidth.

[00406] **FIG. 7** depicts a table 700 showing examples of different traffic or application category types which can be used in implementing network access and content delivery policies. For example, traffic/application categories can include interactive or background, whether a user is waiting for the response, foreground/background application, and whether the backlight is on or off.

[00407] **FIG. 8** depicts a table 800 showing examples of different content category types which can be used in implementing network access and content delivery policies. For

example, content category types can include content of high or low priority, and time critical or non-time critical content/data.

[00408] **FIG. 9** depicts an interaction diagram showing how application (e.g., mobile application) 955 polls having data requests from a mobile device to an application server/content provider 995 over a wireless network can be can be cached on the local proxy 965 and managed by the distributed caching system (including local proxy 965 and the host server 985 (having server cache 935 or caching proxy server 975)).

[00409] In one example, when the mobile application/widget 955 polls an application server/provider 932, the poll can locally be intercepted 934 on the mobile device by local proxy 965. The local proxy 965 can detect that the cached content is available for the polled content in the request and can thus retrieve a response from the local cache to satisfy the intercepted poll 936 without requiring use of wireless network bandwidth or other wireless network resources. The mobile application/widget 955 can subsequently receive a response to the poll from a cache entry 938.

[00410] In another example, the mobile application widget 955 polls the application server/provider 940. The poll is intercepted 942 by the local proxy 965 and detects that cache content is unavailable in the local cache and decides to set up the polled source for caching 944. To satisfy the request, the poll is forwarded to the content source 946. The application server/provider 995 receives the poll request from the application and provides a response to satisfy the current request 948. In 950, the application (e.g., mobile application)/widget 955 receives the response from the application server/provider to satisfy the request.

[00411] In conjunction, in order to set up content caching, the local proxy 965 tracks the polling frequency of the application and can set up a polling schedule to be sent to the host server 952. The local proxy sends the cache set up to the host server 954. The host server 985 can use the cache set up which includes, for example, an identification of the application server/provider to be polled and optionally a polling schedule 956. The host server 985 can now poll the application server/provider 995 to monitor responses to the request 958 on behalf of the mobile device. The application server receives the poll from the host server and responds 960. The host server 985 determines that the same response has been received and polls the application server 995 according to the specified polling schedule 962. The application server/content provider 995 receives the poll and responds accordingly 964.

[00412] The host server 985 detects changed or new responses and notifies the local proxy 965. The host server 985 can additional store the changed or new response in the server cache or caching proxy 968. The local proxy 965 receives notification from the host server 985 that new or changed data is now available and can invalidate the affected cache entries 970. The next time the application (e.g., mobile application)/widget 955 generates the same request for the same server/content provider 972, the local proxy determines that no valid cache entry is available and instead retrieves a response from the server cache 974, for example, through an HTTP connection. The host server 985 receives the request for the new response and sends the response back 976 to the local proxy 965. The request is thus satisfied from the server cache or caching proxy 978 without the need for the mobile device to utilize its radio or to consume mobile network bandwidth thus conserving network resources.

[00413] Alternatively, when the application (e.g., mobile application) generates the same request in step 980, the local proxy 965, in response to determining that no valid cache entry is available, forwards the poll to the application server/provider in step 982 over the mobile network. The application server/provider 995 receives the poll and sends the response back to the mobile device in step 984 over the mobile network. The request is thus satisfied from the server/provider using the mobile network in step 986.

[00414] **FIG. 10** depicts an interaction diagram showing how application 1055 polls for content from an application server/content provider 1095 which employs cache-defeating mechanisms in content identifiers (e.g., identifiers intended to defeat caching) over a wireless network can still be detected and locally cached.

[00415] In one example, when the application (e.g., mobile application)/widget 1055 polls an application server/provider in step 1032, the poll can locally be intercepted in step 1034 on the mobile device by local proxy 1065. In step 1034, the local proxy 1065 on the mobile device may also determine (with some level of certainty and heuristics) that a cache defeating mechanism is employed or may be employed by the server provider.

[00416] The local proxy 1065 can detect that the cached content is available for the polled content in the request and can thus retrieve a response from the local cache to satisfy the intercepted poll 1036 without requiring use of wireless network bandwidth or other wireless network resources. The application (e.g., mobile application)/widget 1055 can subsequently

receive a response to the poll from a cache entry in step 1038 (e.g., a locally stored cache entry on the mobile device).

[00417] In another example, the application (e.g., mobile application) widget 1055 polls the application server/provider 1095 in step 1040. The poll is intercepted in step 1042 by the local proxy 1065 which determines that a cache defeat mechanism is employed by the server/provider 1095. The local proxy 1065 also detects that cached content is unavailable in the local cache for this request and decides to setup the polled content source for caching in step 1044. The local proxy 1065 can then extract a pattern (e.g., a format or syntax) of an identifier of the request and track the polling frequency of the application to setup a polling schedule of the host server 1085 in step 1046.

[00418] To satisfy the request, the poll request is forwarded to the content provider 1095 in step 1048. The application server/provider 1095 receives the poll request from the application and provides a response to satisfy the current request in step 1050. In step 1052, the application (e.g., mobile application)/widget 1055 receives the response from the application server/provider 1095 to satisfy the request.

[00419] In conjunction, in order to setup content caching, the local proxy 1065 caches the response and stores a normalized version of the identifier (or a hash value of the normalized identifier) in association with the received response for future identification and retrieval in step 1054. The local proxy sends the cache setup to the host server 1085 in step 1056. The cache setup includes, for example, the identifier and/or a normalized version of the identifier. In some instances, a modified identifier, different from the normalized identifier, is sent to the host server 1085.

[00420] The host server 1085 can use the cache setup, which includes, for example, an identification of the application server/provider to be polled and optionally a polling schedule in step 1058. The host server 1085 can now poll the application server/provider 1095 to monitor responses to the request in step 1060 on behalf of the mobile device. The application server 1095 receives the poll from the host server 1085 responds in step 1062. The host server 1085 determines that the same response has been received and polls the application server 1095, for example, according to the specified polling schedule and using the normalized or modified identifier in step 1064. The application server/content provider 1095 receives the poll and responds accordingly in step 1066.

[00421] This time, the host server 1085 detects changed or new responses and notifies the local proxy 1065 in step 1068. The host server 1085 can additionally store the changed or new response in the server cache 1035 or caching proxy 1075 in step 1070. The local proxy 1065 receives notification from the host server 1085 that new or changed data is now available and can invalidate the affected cache entries in step 1072. The next time the application (e.g., mobile application)/widget generates the same request for the same server/content provider 1095 in step 1074, the local proxy 1065 determines that no valid cache entry is available and instead retrieves a response from the server cache in step 1076, for example, through an HTTP connection. The host server 1085 receives the request for the new response and sends the response back to the local proxy 1065 in step 1078. The request is thus satisfied from the server cache or caching proxy in step 1080 without the need for the mobile device to utilize its radio or to consume mobile network bandwidth thus conserving network resources.

[00422] Alternatively, when the application (e.g., mobile application) 1055 generates the same request, the local proxy 1065, in response to determining that no valid cache entry is available in step 1084, forwards the poll to the application server provider 1095 in step 1082 over the mobile network. The application server/provider 1095 receives the poll and sends the response back to the mobile device in step 1086 over the mobile network. The request is thus satisfied from the server/provider using the mobile network 1086 in step 1088.

[00423] **FIG. 11** depicts a flow chart illustrating an example process for collecting information about a request and the associated response to identify cacheability and caching the response.

[00424] In process 1102, information about a request and information about the response received for the request is collected. In processes 1104 and 1106, information about the request initiated at the mobile device and information about the response received for the request are used in aggregate or independently to determine cacheability at step 1108. The details of the steps for using request and response information for assessing cacheability are illustrated at flow A as further described in the example of **FIG. 12**.

[00425] In step 1108, if based on flow A it is determined that the response is not cacheable, then the response is not cached in step 1110, and the flow can optionally restart at 1102 to collect information about a request or response to again assess cacheability.

[00426] In step 1108, if it is determined from flow A that the response is cacheable, then in 1112 the response can be stored in the cache as a cache entry including metadata having additional information regarding caching of the response. The cached entry, in addition to the response, includes metadata having additional information regarding caching of the response. The metadata can include timing data including, for example, access time of the cache entry or creation time of the cache entry.

[00427] After the response is stored in the cache, a parallel process can occur to determine whether the response stored in the cache needs to be updated in process 1120. If so, the response stored in the cache of the mobile device is invalided or removed from the cache of the mobile device, in process 1122. For example, relevance or validity of the response can be verified periodically by polling a host server to which the request is directed on behalf of the mobile device. The host server can be polled at a rate determined at the mobile device using request information collected for the request for which the response is cached. The rate is determined from averages of time intervals between previous requests generated by the same client which generated the request.

[00428] The verifying can be performed by an entity that is physically distinct from the mobile device. In one embodiment, the entity is a proxy server coupled to the mobile device and able to communicate wirelessly with the mobile device and the proxy server polls a host server to which the request is directed at the rate determined at the mobile device based on timing intervals between previous requests generated by the same client which generated the request.

[00429] In process 1114, a subsequent request for the same client or application is detected. In process 1116, cache look-up in the local cache is performed to identify the cache entry to be used in responding to the subsequent request. In one embodiment, the metadata is used to determine whether the response stored as the cached entry is used to satisfy the subsequent response. In process 1118, the response can be served from the cache to satisfy a subsequent request. The response can be served in response to identifying a matching cache entry for the subsequent request determined at least in part using the metadata.

[00430] **FIG. 12** depicts a flow chart illustrating an example process for a decision flow to determine whether a response to a request can be cached.

[00431] Process 1202 determines if the request is directed to a blacklisted destination. If so, the response is not cached, in step 1285. If a blacklisted destination is detected, or if the request itself is associated with a blacklisted application, the remainder of the analysis shown in the figure may not be performed. The process can continue to steps 1204 and 1206 if the request and its destination are not blacklisted.

[00432] In process 1204, request characteristics information associated with the request is analyzed. In analyzing the request, in process 1208, the request method is identified and in step 1214, it is determined whether the response can be cached based on the request method. If an uncacheable request is detected, the request is not cached and the process may terminate at process 1285. If the request method is determined to be cacheable, or not uncacheable, then the response can be identified as cacheable or potentially cacheable (e.g., cacheable but subject to the other tests and analysis shown in the figure) at step 1295.

[00433] In process 1210, the size of the request is determined. In process 1216, it is determined whether the request size exceeds a cacheable size. If so, the response is not cached and the analysis may terminate here at process 1285. If the request size does not exceed a cacheable size in step 1216, then the response can be identified as cacheable or potentially cacheable (e.g., cacheable but subject to the other tests and analysis shown in the figure) at step 1295.

[00434] In step 1212, the periodicity information between the request and other requests generated by the same client is determined. In step 1218, it is determined whether periodicity has been identified. If not, the response is not cached and the analysis may terminate here at process 1285. If so, then the response can be identified as cacheable or potentially cacheable (e.g., cacheable but subject to the other tests and analysis shown in the figure) at step 1295.

[00435] In process 1206, the request characteristics information associated with the response received for the request is analyzed.

[00436] In process 1220, the status code is identified and determined whether the status code indicates a cacheable response status code in process 1228. If an uncacheable status code is detected, the request is not cached and the process may terminate at process 1285. If the response status code indicates cacheability, or not uncacheable, then the response can be identified as cacheable or potentially cacheable (e.g., cacheable but subject to the other tests and analysis shown in the figure) at step 1295.

[00437] In process 1222, the size of the response is determined. In process 1230, it is determined whether the response size exceeds a cacheable size. If so, the response is not cached and the analysis may terminate here at process 1285. If the response size does not exceed a cacheable size in step 1230, then the response can be identified as cacheable or potentially cacheable (e.g., cacheable but subject to the other tests and analysis shown in the figure) at step 1295.

[00438] In process 1224, the response body is analyzed. In process 1232, it is determined whether the response contains dynamic content or highly dynamic content. Dynamic content includes data that changes with a high frequency and/or has a short time to live or short time of relevance due to the inherence nature of the data (e.g., stock quotes, sports scores of fast pace sporting events, etc.). If so, the response is not cached and the analysis may terminate here at process 1285. If not, then the response can be identified as cacheable or potentially cacheable (e.g., cacheable but subject to the other tests and analysis shown in the figure) at step 1295.

[00439] Process 1226 determines whether transfer encoding or chunked transfer encoding is used in the response. If so, the response is not cached and the analysis may terminate here at process 1285. If not, then the response can be identified as cacheable or potentially cacheable (e.g., cacheable but subject to the other tests and analysis shown in the figure) at step 1295.

[00440] Not all of the tests described above need to be performed to determined whether a response is cached. Additional tests not shown may also be performed. Note that any of the tests 1208, 1210, 1212, 1220, 1222, 1224, and 1226 can be performed, singly or in any combination to determine cacheability. In some instances, all of the above tests are performed. In some instances, all tests performed (any number of the above tests that are actually performed) need to confirm cacheability for the response to be determined to be cacheable. In other words, in some cases, if any one of the above tests indicate non-cacheability, the response is not cached, regardless of the results of the other tests. In other cases, different criteria can be used to determine which tests or how many tests need to pass for the system to decide to cache a given response, based on the combination of request characteristics and response characteristics.

[00441] **FIG. 13** depicts a flow chart illustrating an example process for determining potential for cacheability based on request periodicity and/or response repeatability.

[00442] In process 1302, requests generated by the client are tracked to detect periodicity of the requests. In process 1306, it is determined whether there are predictable patterns in the timing of the requests. If so, the response content may be cached in process 1395. If not, in process 1308 it is determined whether the request intervals fall within a tolerance level. If so, the response content may be cached in process 1395. If not, the response is not cached in process 1385.

[00443] In process 1304, responses received for requests generated by the client are tracked to detect repeatability in content of the responses. In process 1310, hash values of response bodies of the responses received for the client are examined and in process 1312 the status codes associated with the responses are examined. In process 1314, it is determined whether there is similarity in the content of at least two of the responses using hash values and/or the status codes. If so, the response may be cached in process 1395. If not, the response is not cached in 1385.

[00444] **FIG. 14** depicts a flow chart illustrating an example process for dynamically adjusting caching parameters for a given request or client.

[00445] In process 1402, requests generated by a client or directed to a host are tracked at the mobile device to detect periodicity of the requests. Process 1404 determines if the request intervals between the two or more requests are the same or approximately the same. In process 1406, it is determined that the request intervals between the two or more requests fall within the tolerance level.

[00446] Based on the results of steps 1404 and 1406, the response for the requests for which periodicity is detected is received in process 1408.

[00447] In process 1412, a response is cached as a cache entry in a cache of the mobile device. In process 1414, the host is monitored at a rate to verify relevance or validity of the cache entry, and simultaneously, in process 1416, the response can be served from the cache to satisfy a subsequent request.

[00448] In process 1410, a rate to monitor a host is determined from the request interval, using, for example, the results of processes 1404 and/or 1406. In process 1420, the rate at which the given host is monitored is set to verify relevance or validity of the cache entry for the requests. In process 1422, a change in request intervals for requests generated by the client

is detected. In process 1424, a different rate is computed based on the change in request intervals. The rate at which the given host is monitored to verify relevance or validity of the cache entry for the requests is updated in step 1420.

[00449] **FIG. 15** depicts a flow chart illustrating example processes for application and/or traffic (data) categorization while factoring in user activity and expectations for implementation of network access and content delivery policies.

[00450] In process 1502, a system or server detects that new or changed data is available to be sent to a mobile device. The data, new, changed, or updated, can include one or more of, IM presence updates, stock ticker updates, weather updates, mail, text messages, news feeds, friend feeds, blog entries, articles, documents, any multimedia content (e.g., images, audio, photographs, video, etc.), or any others that can be sent over HTTP or wireless broadband networks, either to be consumed by a user or for use in maintaining operation of an end device or application.

[00451] In process 1504, the application to which the new or changed data is directed is identified. In process 1506, the application is categorized based on the application. In process 1508, the priority or time criticality of the new or changed data is determined. In process 1510, the data is categorized. Based on the information determined from the application and/or priority/time-sensitivity of the relevant data, any or all of a series of evaluations can be performed to categorize the traffic and/or to formulate a policy for delivery and/or powering on the mobile device radio.

[00452] For example, using the identified application information, in process 1512, it is determined whether the application is in an active state interacting with a user on a mobile device. In process 1514, it is determined if the application is running in the foreground on the mobile device.

[00453] If the answer is 'Yes' to any number of the test of processes 1512 or 1514, the system or server can then determine that the new or changed data is to be sent to the mobile device in step 1526, and sent without delay. Alternatively, the process can continue at flow 'C' where the timing, along with other transmission parameters such as network configuration, can be selected, as further illustrated in the example of **FIG. 16C**. If the answer is 'No' to the tests of 1512 or 1514, the other test can be performed in any order. As long as one of the tests

1512 or 1514 is 'Yes,' then the system or server having the data can proceed to step 1526 and/or flow 'C.'

[00454] If the answer is 'No' to the tests 1512 and 1514 based on the application or application characteristics, then the process can proceed to step 1524, where the sending of the new or changed data is suppressed, at least on a temporary basis. The process can continue in flow 'A' for example steps for further determining the timing of when to send the data to optimize network use and/or device power consumption, as further described in the example of flow chart in **FIG. 16A**.

[00455] Similarly, in process 1516, it is determined whether the application is running in the background. If so, the process can proceed to step 1524 where the sending of the new or changed data is suppressed. However, even if the application is in the background state, any of the remaining tests can be performed. For example, even if an application is in the background state, new or changed data may still be sent if of a high priority or is time critical.

[00456] Using the priority or time sensitivity information, in process 1518, it is determined whether the data is of high priority 1518. In process 1520, it is determined whether the data is time critical. In process 1522, it is determined whether a user is waiting for a response that would be provided in the available data.

[00457] If the answer is 'Yes' to any number of the test of processes 1518, 1520, or 1522, the system or server can then determine that the new or changed data is to be sent to the mobile device in step 1526, and sent without delay. Alternatively, the process can continue at flow 'C' where the timing, along with other transmission parameters such as a network configuration, can be selected, as further illustrated in the example of **FIG. 16C**. If the answer is 'No' to any of these tests, the other test can be performed in any order. As long as one of the tests 1518, 1520, or 1522 is 'Yes,' then the system or server having the data can proceed to step 1526 and/or flow 'C.'

[00458] If the answer is 'No' to one or more of the tests 1518, 1520, or 1522, then the process can proceed to step 1524, where the sending of the new or changed data is suppressed, at least on a temporary basis. The process can continue in flow 'A' for example steps for further determining the timing of when to send the data to optimize network use and/or device power consumption. The process can continue to step 1524 with or without the other tests being performed if one of the tests yields a 'No' response.

[00459] The determined application category in step 1504 can be used in lieu of or in conjunction with the determined data categories in step 1510. For example, the new or changed data that is of a high priority or is time critical can be sent at step 1526 even if the application in the foreground state but not actively interacting with the user on the mobile device or if the application is not in the foreground, or in the background.

[00460] Similarly, even if the user is not waiting for a response which would be provided in the new or change data (in step 1522), the data can be sent to the mobile device 1526 if the application is in the foreground, or if the data is of high priority or contains time critical content.

[00461] In general, the suppression can be performed at the content source (e.g., originating server/content host of the new or changed data), or at a proxy server. For example, the proxy server may be remote from the recipient mobile device (e.g., able to wirelessly connect to the receiving mobile device). The proxy server may also be remote from the originating server/content host. Specifically, the logic and intelligence in determining whether the data is to be sent or suppressed can exist on the same server or be the same entity as the originator of the data to be sent or partially or wholly remote from it (e.g., the proxy is able to communicate with the content originating server).

[00462] In one embodiment, the waiting to transfer the data is managed by a local proxy on the mobile device which is able to wirelessly communicate with a recipient server (e.g., the host server for the mobile application or client). The local proxy on the mobile device can control the radio use on the mobile device for transfer of the data when the time period has elapsed, or when additional data to be sent is detected.

[00463] **FIG. 16A** depicts a flow chart illustrating example processes for handling traffic which is to be suppressed at least temporarily determined from application/traffic categorization.

[00464] For example, in process 1602, a time period is elapsed before the new or change data is transmitted in step 1606. This can be performed if the data is of low priority or is not time critical, or otherwise determined to be suppressed for sending (e.g., as determined in the flow chart of **FIG. 15**). The time period can be set by the application, the user, a third party, and/or take upon a default value. The time period may also be adapted over time for specific types of applications or real-time network operating conditions. If the new or changed data to

be sent is originating from a mobile device, the waiting to transfer of the data until a time period has elapsed can be managed by a local proxy on the mobile device, which can communicate with the host server. The local proxy can also enable or allow the use radio use on the mobile device for transfer of the data when the time period has elapsed.

[00465] In some instances, the new or changed data is transmitted in 1606 when there is additional data to be sent, in process 1604. If the new or changed data to be sent is originating from a mobile device, the waiting to transfer of the data until there is additional data to be sent, can be managed by a local proxy on the mobile device, which can communicate with the host server. The local proxy can also enable or allow the use radio use on the mobile device for transfer of the data when there is additional data to be sent, such that device resources can be conserved. Note that the additional data may originate from the same mobile application/client or a different application/client. The additional data may include content of higher priority or is time critical. The additional data may also be of same or lower priority. In some instances, a certain number of non priority, or non time-sensitive events may trigger a send event.

[00466] If the new or changed data to be sent is originating from a server (proxy server or host server of the content), the waiting to transfer of the data until a time period has elapsed or waiting for additional data to be sent, can be managed by the proxy server which can wirelessly communicate with the mobile device. In general, the proxy server waits until additional data is available for the same mobile device before sending the data together in a single transaction to minimize the number of power-ons of device battery and to optimize network use.

[00467] **FIG. 16B** depicts a flow chart illustrating an example process for selection of a network configuration for use in sending traffic based on application and/or traffic (data) categorization.

[00468] In process 1608, an activity state of an application on the mobile device is detected for which traffic is directed to or originated from is detected. In parallel or in lieu of activity state, a time criticality of data contained in the traffic to be sent between the mobile device and the host server can be determined, in process 1610. The activity state can be determined in part or in while, by whether the application is in a foreground or background state on the mobile device. The activity state can also be determined by whether a user is interacting the application.

[00469] Using activity state and/or data characteristics, when it has determined from that the data is to be sent to the mobile device in step 1612 of **FIG. 15**, the process can continue to step 3006 for network configuration selection.

[00470] For example, in process 1614, a generation of wireless standard is selected. The generation of wireless standard which can be selected includes 2G or 2.5G, 3G, 3.5G, 3G+, 3GPP, LTE, or 4G, or any other future generations. For example, slower or older generation of wireless standards can be specified for less critical transactions or traffic containing less critical data. For example, older standards such as 2G, 2.5G, or 3G can be selected for routing traffic when one or more of the following is detected, the application is not interacting with the user, the application is running in the background on the mobile device, or the data contained in the traffic is not time critical. Newer generations such as can be specified for higher priority traffic or transactions. For example, newer generations such as 3G, LTE, or 4G can be specified for traffic when the activity state is in interaction with a user or in a foreground on the mobile device.

[00471] In process 1616, the access channel type can be selected. For example, forward access channel (FACH) or the dedicated channel (DCH) can be specified. In process 1618, a network configuration is selected based on data rate or data rate capabilities. For example, a network configuration with a slower data rate can be specified for traffic when one or more of the following is detected, the application is not interacting with the user, the application is running in the background on the mobile device, or the data contained in the traffic is not time critical

[00472] In process 1620, a network configuration is selected by specifying access points. Any or all of the steps 1614, 1616, 1618, and 1620 can be performed or in any combination in specifying network configurations.

[00473] **FIG. 16C** depicts a flow chart illustrating an example process for implementing network access and content delivery policies based on application and/or traffic (data) categorization.

[00474] In process 1634, an activity state of an application on a mobile device to which traffic is originated from or directed to is detected. For example, the activity state can be determined by whether the application is in a foreground or background state on the mobile

device. The activity state can also be determined by whether a user is expecting data contained in the traffic directed to the mobile device.

[00475] In process 1636, a time criticality of data contained in the traffic to be sent between the mobile device and the host server is detected. For example, when the data is not time critical, the timing with which to allow the traffic to pass through can be set based on when additional data needs to be sent. Therefore, the traffic can be batched with the other data so as to conserve network and/or device resources.

[00476] The application state and/or data characteristics can be used for application categorization and/or data categorization to determine whether the traffic resulting therefrom is to be sent to the mobile device or suppressed at least on a temporary basis before sending, as illustrated in the flow chart shown in the example of **FIG. 15**.

[00477] Continuing at flow C after a determination has been made to send the traffic, the parameters relating to how and when the traffic is to be sent can be determined. For example, in process 1638, a timing with which to allow the traffic to pass through, is determined based on the activity state or the time criticality.

[00478] In process 1640, radio use on the mobile device is controlled based on the timing with which the traffic is allowed to pass through. For example, for traffic initiated from the mobile device, a local proxy can residing on the mobile device can control whether the radio is to be turned on for a transaction, and if so, when it is to be turned on, based on transaction characteristics determined from application state, or data priority/time-sensitivity.

[00479] In process 1642, a network configuration in the wireless network is selected for use in passing traffic to and/or from the mobile device. For example, a higher capacity or data rate network (e.g., 3G, 3G+, 3.5G, LTE, or 4G networks) can be selected for passing through traffic when the application is active or when the data contained in the traffic is time critical or is otherwise of a higher priority/importance.

[00480] **FIG. 17** depicts a flow chart illustrating an example process for network selection based on mobile user activity or user expectations.

[00481] In process 1702, the backlight status of a mobile device is detected. The backlight status can be used to determine or infer information regarding user activity and/or user

expectations. For example, in process 1704, user interaction with an application on a mobile device is detected and/or in process 1706, it is determined that a user is expecting data contained in traffic directed to the mobile device, if the backlight is on.

[00482] The user interaction 1704 and/or user expectation 1706 can be determined or inferred via other direct or indirect cues. For example, device motion sensor, ambient light, data activity, detection of radio activity and patterns, call processing, etc. can be used alone or in combination to make an assessment regarding user activity, interaction, or expectations.

[00483] In process 1708, an activity state of an application on the mobile device for which traffic is originated from or directed to, is determined. In one embodiment, the activity state of the application is determined by user interaction with the application on the mobile device and/or by whether a user is expecting data contained in the traffic directed to the mobile device.

[00484] In process 1710, 3G, 4G, or LTE network is selected for use in sending traffic between a mobile device and a host server in the wireless network. Other network configurations or technologies can be selected as well, including but not limited to 2.5G GSM/GPRS networks, EDGE/EGPRS, 3.5G, 3G+, turbo 3G, HSDPA, etc. For example, a higher bandwidth or higher capacity network can be selected when user interaction is detected with an application requesting to access the network. Similarly, if it can be determined or inferred with some certainty that the user may be expecting data contained in traffic requesting network access, a higher capacity or higher data rate network may be selected as well.

[00485] The activity state can also be determined by whether data contained in the traffic directed to the mobile device responds to foreground activities in the application. For applications which are in the foreground, a higher capacity (e.g., 3.5G, 4G, or LTE) network may be selected for use in carrying out the transaction.

[00486] The activity state can be determined via device parameters such as the backlight status of the mobile device or any other software or hardware based device sensors including but not limited to, resistive sensors, capacitive sensors, light detectors, motion sensors, proximity sensors, touch screen sensors, etc. The network configuration which is selected for use can be further based on a time criticality and/or priority of data contained in the traffic to be sent between the mobile device and the host server.

[00487] **FIG. 18** depicts a data timing diagram 1800 showing an example of detection of periodic request which may be suitable for caching.

[00488] In the example shown, a first request from a client/application on a mobile device is detected at time 1:00 (t1). At this time, a cache entry may be created in step 1802. At time 2:00 (t2), the second request is detected from the same client/application, and the cache entry that was created can now be updated with the detected interval of 1 hour between time t2 and t1 at step 1804. The third request from the same client is now detected at time t3 = 3:00, and it can now be determined that a periodic request is detected in step 1806. The local proxy can now cache the response and send a start poll request specifying the interval (e.g., 1 hour in this case) to the proxy server.

[00489] The timing diagram further illustrates the timing window between 2:54 and 3:06, which indicates the boundaries of a window within which periodicity would be determined if the third request is received within this time frame 1810. The timing window 1808 between 2:54 and 3:06 corresponds to 20% of the previous interval and is the example tolerance shown. Other tolerances may be used, and can be determined dynamically or on a case by case (application by application) basis.

[00490] **FIG. 19** depicts a data timing diagram 1900 showing an example of detection of change in request intervals and updating of server polling rate in response thereto.

[00491] At step 1902, the proxy determines that a periodic request is detected, the local proxy caches the response and sets the polling request to the proxy server, and the interval is set to 1 hour at the 3rd request, for example. At time t4=3:55, the request is detected 55 minutes later, rather than 1 hour. The interval of 55 minutes still fits in to the window 1904 given a tolerance of 20%. However, at step 1906, the 5th request is received at time t5 = 4:50, which no longer fits within the tolerance window set determined from the interval between the 1st and second, and second and third requests of 1 hour. The local proxy now retrieves the resource or response from the proxy server, and refreshes the local cache (e.g., cache entry not used to serve the 5th request). The local proxy also resends a start poll request to the proxy server with an updated interval (e.g., 55 minutes in the example) and the window defined by the tolerance, set by example to 20%, now becomes 11 minutes, rather than 12 minutes.

[00492] Note that in general, the local proxy notifies the proxy server with an updated polling interval when an interval changes is detected and/or when a new rate has been

determined. This is performed, however, typically only for background application requests or automatic/programmatic refreshes (e.g., requests with no user interaction involved). In general, if the user is interacting with the application in the foreground and causing out of period requests to be detected, the rate of polling or polling interval specified to the proxy server is typically not update, as illustrated in **FIG. 20**. **FIG. 20** depicts a data timing diagram 2000 showing an example of serving foreground requests with cached entries.

[00493] For example, between the times of t = 3:00 and 3:30, the local proxy detects 1st and 2nd foreground requests at t = 3:10 and t = 3:20. These foreground requests are outside of the periodicity detected for background application or automatic application requests. The response data retrieved for the foreground request can be cached and updated, however, the request interval for foreground requests are not sent to the server in process 2008.

[00494] As shown, the next periodic request detected from the application (e.g., a background request, programmatic/automatic refresh) at t=4:00, the response is served from the cache, as is the request at t=5:00.

[00495] **FIG. 21** depicts a data timing diagram 2100 showing an example of a non-optimal effect of cache invalidation occurring after outdated content has been served once again to a requesting application.

[00496] Since the interval of proxy server polls is set to approximately the same interval at which the application (e.g., mobile application) is sending requests, it is likely the case that the proxy server typically detects changed content (e.g., at t=5:02) after the cached entry (now outdated) has already been served for a request (e.g., to the 5th request at t=5:00). In the example shown, the resource updates or changes at t=4:20 and the previous server poll which occurs at t = 4:02 was not able to capture this change until the next poll at 5:02 and sends a cache invalidation to the local proxy at 2110. Therefore, the local cache does not invalidate the cache at some time after the 5th request at time t=5:00 has already been served with the old content. The fresh content is now not provided to the requesting application until the 6th request at t = 6:00, 1 period later at process 2106.

[00497] To optimize caching performance and to resolve this issue, the local proxy can adjust time setup by specifying an initial time of request, in addition to the polling interval to the proxy server. The initial time of request here is set to some time before (e.g., a few minutes) the request actually occurred such that the proxy server polls occur slightly before

actual future application requests. This way, the proxy can pick up any changes in responses in time to be served to the subsequent application request.

[00498] **FIG. 22** depicts a data timing diagram 2200 showing cache management and response taking into account the time-to-live (TTL) set for cache entries.

[00499] In one embodiment, cached response data in the local cache specifies the amount of time cache entries can be stored in the local cache until it is deleted or removed.

[00500] The time when a response data in a given cache entry is to be removed can be determined using the formula: <response data_cache time> + <TTL>, as shown at t = 3:00, the response data is automatically removed after the TTL has elapsed due to the caching at step 2212 (e.g., in this example, 24 hours after the caching at step 2212). In general the time to live (TTL) applies to the entire cache entry (e.g., including both the response data and any metadata, which includes information regarding periodicity and information used to compute periodicity). In one embodiment, the cached response data TTL is set to 24 hours by default or some other value (e.g., 6 hours, 12 hours, 48 hours, etc.). The TTL may also be dynamically adjustable or reconfigured by the admin/user and/or different on a case-by-case, device, application, network provider, network conditions, operator, and/or user-specific basis.

[00501] **FIG. 23** depicts a flow chart illustrating an example flow for generating state map for the device to determine session state at a device for determining whether activity occurred when expected or predicted.

[00502] In process 2302, information or application usage pattern of a user at a device is gathered. In process 2304, a state map is generated for the device to determine session state at a device. In one embodiment, the state map can also be determined from information gathered in the wireless network. the information gathered in the wireless network can include one or more of network capacity, network congestion, network bandwidth, network generation or network outage. In one embodiment, the state map can be stored in a local proxy on the device.

[00503] The state map can be used to maintain the session state such that messages need not be transmitted in a wireless network to maintain or determine the session state of the device with another entity in the wireless network. The other entity can be the host server or the proxy server (e.g., the proxy server of the distributed proxy and/or cache system of **FIG. 1B-FIG. 1C**, **FIG. 3A-FIG. 3B**, or **FIG. 5A-FIG. 5D**) in the wireless network remote from the device,

or a content source/application server 110 of **FIG. 1B-FIG. 1C**. For example, the other entity can include a content host to receive the request for information which would have been received as the result of the activity.

[00504] In process 2306, the state map is used to assess if or when activity is expected or predicted to occur at the device. If so, and in process 2310, it is determined that activity did not occur when expected or predicted, then in process 2312, the device sends a request for information.

[00505] Alternatively or in combination with process 2312, the device can reset a cache state in process 2314. For example, the cache state is maintained between the proxy server and the device and resets when the activity does not occur when expected or predicted. In some instances, the if and when the activity is expected to occur is represented by a quantitative value, for example, the quantitative value includes a threshold value of a lack of or no communication to or from the device as determined from the state map.

[00506] In one embodiment, the threshold value is specific to a user of the device or habits of the user of the device. The threshold value can be specific to a mobile application from which the activity is expected to occur. The threshold value can also be specific to communication patterns of the mobile application. The communication patterns can include specific communication patterns during foreground mode or background mode. In process 2308, it is determined whether a threshold value of a lack of or no communication been reached or exceeded. In one embodiment, the device has access to thereon multiple mobile applications, and different threshold values area associated with each of the multiple mobile applications.

[00507] **FIG. 24** depicts a flow chart illustrating an example flow for conserving resources of the network when maintaining or conveying the session state of the mobile device.

[00508] In process 2402, state of the mobile device from information specific to the mobile network to which the mobile device is connected is determined. In process 2404, the state status of the mobile device is used to assess if or when communication is expected or predicted to occur at the mobile device. In one embodiment, the state status is determined from information specific to the mobile network to which the mobile device is connected.

[00509] If the communication does not occur when expected or predicted, the device can send a request for information which would have been received as a result of the activity. Alternatively, if the communication does not occur when expected or predicted, the device can reset a cache state on the device. In one embodiment, the reset of the cache state on the device, includes invalidation of cache entries that are cached from long poll requests. In one embodiment, the reset of the cache state on the device, includes invalidation of cache entries that are cached in a local cache on the device; wherein, the cache entries include responses from a source that employ a cache defeat mechanism.

[00510] In process 2406, the session state is maintained such that health-status messages need not be transmitted in the network. In process 2408, resources of the network is thus conserved when maintaining or conveying the session state of mobile device.

[00511] **FIG. 25** depicts a flow chart illustrating an example flow for preserving session state information among a first device coupled to a second device via a wireless network.

[00512] In process 2502, state information of a first device is aggregated. The state information can be used to assess health status of the first device and enables the session state information to be preserved between the first device and a second device without a need for transmission of messages in the wireless network. In process 2504, a time frame as to when correspondence or data should be received from the second device by the first device based is determined or predicted on the state information of the first device. In process 2506, it is determined generally how frequently and when data is requested or received at the first mobile device, either in the foreground or in the background. In one embodiment, the time frame is represented by a tolerable threshold of lack of communication for the first device. In one embodiment, the tolerable threshold can be computed from the state information.

[00513] In process 2508, a tolerable threshold of lack of communication for the first device is computed to represent the time frame as to when correspondence or data should be received from the second device. In process 2510, an amount of time from which a previous correspondence or data was received from the second device is tracked or computed. In process 2512, health status of the first device is assessed based on the amount of time compared to the tolerable threshold. In process 2514, based on the assessment of the health status, a request for information which would have been received is sent as a result of the activity or a cache state is reset on the device,

[00514] **FIG. 26** depicts a flow chart illustrating example processes for gathering state information used in assessing or determining device or session health state.

[00515] In process 2602, state information is aggregated or gathered via one or more of steps 2604-2612. For example, in step 2602, behavior of a user of the first device is determined. In process 2604, behavior of a second user of the first device is determined. In step 2606, the time of day or day of week or other timing parameters is determined. In step 2608, wireless network information is determined. In step 2610, communication patterns of a mobile application is determined. In one embodiment, the state information can also include, one or more of, what applications are installed on the first device and what applications are accessed on the first device, and what applications are running or in the foreground on the first device.

[00516] The above information can be used alone or in any combination for determining state status or state information of a device or one or more devices for use in assessing device or session state health. The state status or state information can optionally be used in generating a state map from which device or session state health can be derived or determined such that session state can be maintain cross a network among multiple network entities without the need to transmit health-status messages in the network (e.g., a mobile network, WIFI network or other networks).

[00517] **FIG. 27** shows a diagrammatic representation of a machine in the example form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed.

[00518] In alternative embodiments, the machine operates as a standalone device or may be connected (e.g., networked) to other machines. In a networked deployment, the machine may operate in the capacity of a server or a client machine in a client-server network environment, or as a peer machine in a peer-to-peer (or distributed) network environment.

[00519] The machine may be a server computer, a client computer, a personal computer (PC), a user device, a tablet PC, a laptop computer, a set-top box (STB), a personal digital assistant (PDA), a cellular telephone, an iPhone, an iPad, a Blackberry, a processor, a telephone, a web appliance, a network router, switch or bridge, a console, a hand-held console, a (hand-held) gaming device, a music player, any portable, mobile, hand-held device, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine.

[00520] While the machine-readable medium or machine-readable storage medium is shown in an exemplary embodiment to be a single medium, the term "machine-readable medium" and "machine-readable storage medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database and/or associated caches and servers) that store the one or more sets of instructions. The term "machine-readable medium" and "machine-readable storage medium" shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the presently disclosed technique and innovation.

[00521] In general, the routines executed to implement the embodiments of the disclosure may be implemented as part of an operating system or a specific application, component, program, object, module or sequence of instructions referred to as "computer programs." The computer programs typically comprise one or more instructions set at various times in various memory and storage devices in a computer that, when read and executed by one or more processing units or processors in a computer, cause the computer to perform operations to execute elements involving the various aspects of the disclosure.

[00522] Moreover, while embodiments have been described in the context of fully functioning computers and computer systems, those skilled in the art will appreciate that the various embodiments are capable of being distributed as a program product in a variety of forms, and that the disclosure applies equally regardless of the particular type of machine or computer-readable media used to actually effect the distribution.

[00523] Further examples of machine-readable storage media, machine-readable media, or computer-readable (storage) media include but are not limited to recordable type media such as volatile and non-volatile memory devices, floppy and other removable disks, hard disk drives, optical disks (e.g., Compact Disk Read-Only Memory (CD ROMS), Digital Versatile Disks, (DVDs), etc.), among others, and transmission type media such as digital and analog communication links.

[00524] Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to." As used herein, the terms "connected," "coupled," or any variant thereof,

means any connection or coupling, either direct or indirect, between two or more elements; the coupling of connection between the elements can be physical, logical, or a combination thereof. Additionally, the words "herein," "above," "below," and words of similar import, when used in this application, shall refer to this application as a whole and not to any particular portions of this application. Where the context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number respectively. The word "or," in reference to a list of two or more items, covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list.

[00525] The above detailed description of embodiments of the disclosure is not intended to be exhaustive or to limit the teachings to the precise form disclosed above. While specific embodiments of, and examples for, the disclosure are described above for illustrative purposes, various equivalent modifications are possible within the scope of the disclosure, as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative embodiments may perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified to provide alternative or sub-combinations. Each of these processes or blocks may be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks may instead be performed in parallel, or may be performed at different times. Further any specific numbers noted herein are only examples: alternative implementations may employ differing values or ranges.

[00526] The teachings of the disclosure provided herein can be applied to other systems, not necessarily the system described above. The elements and acts of the various embodiments described above can be combined to provide further embodiments.

[00527] Any patents and applications and other references noted above, including any that may be listed in accompanying filing papers, are incorporated herein by reference. Aspects of the disclosure can be modified, if necessary, to employ the systems, functions, and concepts of the various references described above to provide yet further embodiments of the disclosure.

[00528] These and other changes can be made to the disclosure in light of the above Detailed Description. While the above description describes certain embodiments of the

disclosure, and describes the best mode contemplated, no matter how detailed the above appears in text, the teachings can be practiced in many ways. Details of the system may vary considerably in its implementation details, while still being encompassed by the subject matter disclosed herein. As noted above, particular terminology used when describing certain features or aspects of the disclosure should not be taken to imply that the terminology is being redefined herein to be restricted to any specific characteristics, features, or aspects of the disclosure with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the disclosure to the specific embodiments disclosed in the specification, unless the above Detailed Description section explicitly defines such terms. Accordingly, the actual scope of the disclosure encompasses not only the disclosed embodiments, but also all equivalent ways of practicing or implementing the disclosure under the claims.

[00529] While certain aspects of the disclosure are presented below in certain claim forms, the inventors contemplate the various aspects of the disclosure in any number of claim forms. For example, while only one aspect of the disclosure is recited as a means-plus-function claim under 35 U.S.C. §112, ¶6, other aspects may likewise be embodied as a means-plus-function claim, or in other forms, such as being embodied in a computer-readable medium. (Any claims intended to be treated under 35 U.S.C. §112, ¶6 will begin with the words "means for.") Accordingly, the applicant reserves the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the disclosure.

Claims

What is claimed is:

1. A method which uses a state map to determine session state at a device, the method comprising:

using the state map to assess if or when activity is expected or predicted to occur at the device;

wherein, if the activity does not occur when expected or predicted, the device sends a request for information which would have been received as a result of the activity or resets a cache state on the device.

2. The method of claim 1, wherein, the state map is used to maintain the session state such that messages need not be transmitted in a wireless network to maintain or determine the session state of the device with another entity in the wireless network.

3. The method of claim 1 wherein, the state map is determined from information gathered at the device.

4. The method of claim 3, wherein, the information includes device or application usage patterns of a user of the device.

5. The method of claim 3, wherein, the state map is determined from information gathered in the wireless network.

6. The method of claim 5, wherein, the information gathered in the wireless network includes one or more of network capacity, network congestion, network bandwidth, network generation or network outage.

7. The method of claim 1, wherein, the state map is stored in a local proxy on the device.

8. The method of claim 2, wherein, the other entity is a proxy server in a distributed proxy or cache system in the wireless network remote from the device.

9. The method of claim 8, wherein, the cache state is maintained between the proxy server and the device and resets when the activity does not occur when expected or predicted.

10. The method of claim 8, wherein, the other entity includes a content host to receive the request for information which would have been received as the result of the activity.

11. The method of claim 1, wherein, the if and when the activity is expected to occur is represented by a quantitative value.

12. The method of claim 11, wherein, the quantitative value includes a threshold value of a lack of or no communication to or from the device as determined from the state map.

13. The method of claim 1, wherein, the state map is specific to the device.

14. The method of claim 12, wherein, the threshold value is specific to a user of the device or habits of the user of the device.

15. The method of claim 12, wherein, the threshold value is specific to a mobile application from which the activity is expected to occur.

16. The method of claim, 12, wherein, the threshold value is specific to communication patterns of the mobile application.

17. The method of claim 16, wherein, the communication patterns include communication patterns during foreground mode.

18. The method of claim 16, wherein, the communication patterns include communication patterns during background mode.

19. The method of claim 15, wherein, the device has access to thereon multiple mobile applications, and different threshold values are associated with each of the multiple mobile applications.

20. A machine readable storage medium having stored thereon instructions which when executed by a processor causes a method to be performed by a mobile device to maintain session state information of a mobile device without a need to transmit health-status messages in a network, the method, comprising

using the state status of the mobile device to assess if or when communication is expected or predicted to occur at the mobile device;

wherein, the state status is used to maintain the session state such that health-status messages need not be transmitted in the network to maintain or convey the session state of mobile device such that resources of the network is conserved;

wherein, the state status is determined from information specific to the mobile network to which the mobile device is connected.

21. The method of claim 20, wherein, if the communication does not occur when expected or predicted, the device sends a request for information which would have been received as a result of the activity.

22. The method of claim 20, wherein, if the communication does not occur when expected or predicted, the device resets a cache state on the device.

23. The method of claim 22, wherein, the reset of the cache state on the device, includes invalidation of cache entries that are cached from long poll requests.

24. The method of claim 22, wherein, the reset of the cache state on the device, includes invalidation of cache entries that are cached in a local cache on the device; wherein, the cache entries include responses from a source that employ a cache defeat mechanism.

25. A mobile device which maintains state information without a need to transmit health-status messages in a network, the mobile device, comprising:

a radio;

a processing unit; a memory unit having instructions stored thereon which when executed by the processing unit, causes the mobile device to:

use a state status of the mobile device to assess if or when communication is expected or predicted to occur at the mobile device;

wherein, the state status is used to maintain the session state such that health-status messages need not be transmitted in the network to maintain or convey the session state of mobile device such that resources of the network is conserved.

26. The mobile device of claim 25, wherein, the state status is determined from information specific to the mobile network to which the mobile device is connected.

27. The mobile device of claim 25, wherein, the state status is determined from what applications are accessed via the mobile device.

28. The mobile device of claim 25, wherein, the state status is determined from what applications are running.

29. The mobile device of claim 25, wherein, the state status is determined from what applications are in the foreground or in the background on the mobile device.

30. A system to maintain session state without a need to transmit health-status messages in a network, the system, comprising

means for, using a state status of a session state to assess if or when communication is expected or predicted to occur at an entity in the network;

wherein, the if and when the activity is expected to occur is represented by a quantitative value;

wherein, the quantitative value includes a threshold value of a lack of or no communication in the network;

wherein, the state status is used to maintain the session state such that health-status messages need not be transmitted in the network to maintain or convey the session state such that resources of the network is conserved;

wherein, the state status is determined from information specific to the network.

31. A method of preserving session state information among a first device coupled to a second device via a wireless network, the method, including:

aggregating state information of the first device used to determine or predict a time frame as to when correspondence or data should be received from the second device;

wherein the time frame is represented by a tolerable threshold of lack of communication for the first device;

wherein, the state information enables the session state information to be preserved between the first device and the second device without a need for transmission of messages in the wireless network.

32. The method of claim 31, wherein, the state information is used to assess health status of the first device; wherein, the tolerable threshold is computed from the state information.

33. The method of claim 31, wherein, the state information is used to determine generally how frequently and when data is requested or received at the first mobile device, either in the foreground or in the background

34. The method of claim 31, wherein, the state information includes, behavior of a user of the first device.

35. The method of claim 31, wherein, the state information includes, behavior of a second different user of the first device.

36. The method of claim 31, wherein, the state information incudes, one or more of, time of day or day of week.

37. The method of claim 31, wherein, the state information includes, what applications are installed on the first device.

38. The method of claim 31, wherein, the state information includes, what applications are accessed or in use on the first device.

39. The method of claim 31, wherein, the state information includes, what applications are running.

40. The method of claim 31, wherein, the state information includes, what applications are in the background on the first device.

41. The method of claim 31, wherein, the state information includes, what applications are in the foreground on the first device.

42. A system of preserving state information with multiple mobile devices in a wireless network, the system, comprising:

a proxy server in the wireless network;

wherein, the proxy server:

aggregates state information each of the multiple devices used to determine or predict time frames as to when correspondence or data should be communicated between each of the multiple devices and the proxy server;

wherein the time frame is represented by a tolerable threshold of lack of communication for each of the multiple devices, and is specific to each of the multiple devices.

43. The system of claim 42, wherein, the state information enables the state information to be preserved among the multiple mobile devices and the proxy server the without a need for transmission of messages in the wireless network.

44. The system of claim 42, wherein, the proxy server is distinct from a component of a network operator of the wireless network and in communication with the network operator.

45. The system of claim 42, wherein, the proxy server is a component of a network operator of the wireless network.

46. The system of claim 42, wherein, the state information includes, one or more of, what applications are installed or accessed on the multiple devices.

47. A mobile device which maintains state information without a need to transmit status messages in a mobile network, the mobile device, comprising:

a radio;

a processing unit; a memory unit having instructions stored thereon which when executed by the processing unit, causes the mobile device to:

use state information to assess if or when activity is expected or predicted to occur at the mobile device;

wherein, if the activity does not occur when expected or predicted, the mobile device sends a request for information which would have been received as a result of the activity or resets a cache state on the device.

48. The mobile device of claim 47, wherein, the state information is determined from information specific to the mobile network to which the mobile device is connected.

49. The mobile device of claim 47, wherein, the information includes device or application usage patterns of a user of the device.

50. The mobile device of claim 47, wherein, the information gathered in the wireless network includes one or more of network capacity, network congestion, network bandwidth, network generation or network outage.

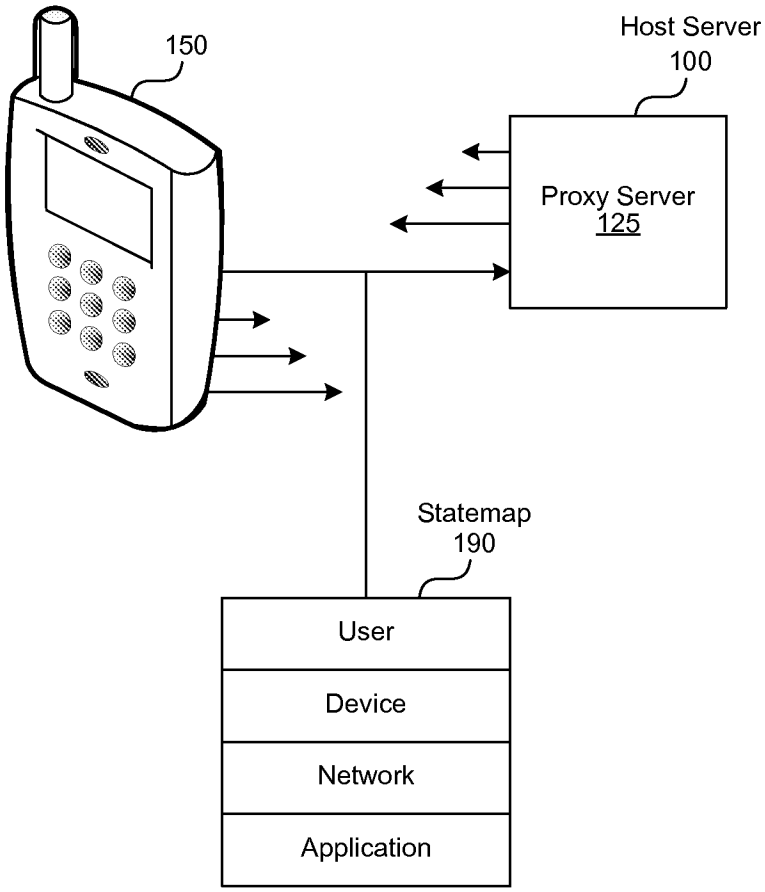

150
Host Server
100
Proxy Server
125
Statemap
190
User
Device
Network
Application

FIG. 1A

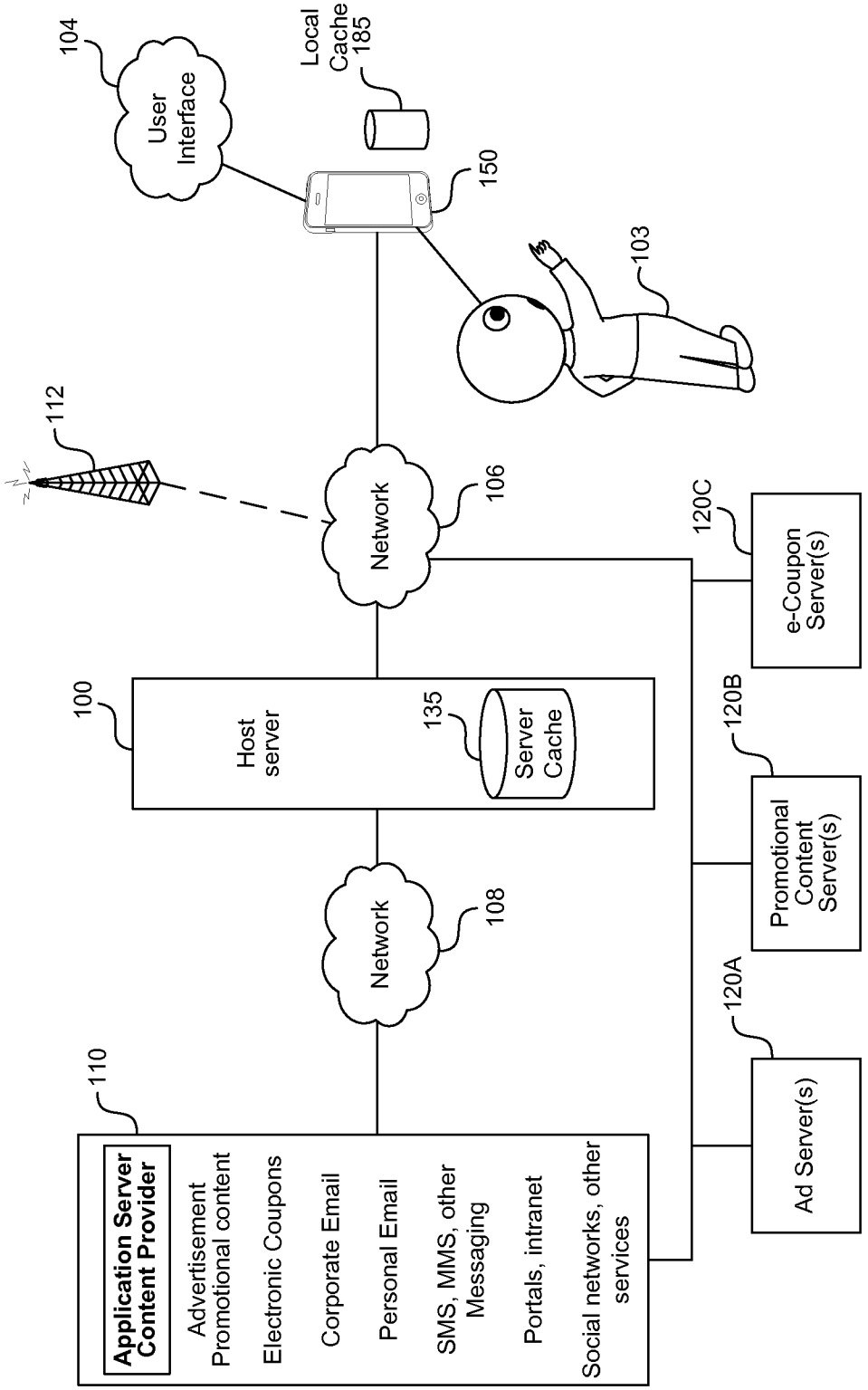

104
User Interface
Local Cache 185
150
103
112
Network
106
120C
e-Coupon Server(s)
100
Host server
135
Server Cache
120B
Promotional Content Server(s)
Network
108
120A
Ad Server(s)
110
Application Server Content Provider
Advertisement Promotional content
Electronic Coupons
Corporate Email
Personal Email
SMS, MMS, other Messaging
Portals, intranet
Social networks, other services

FIG. 1B

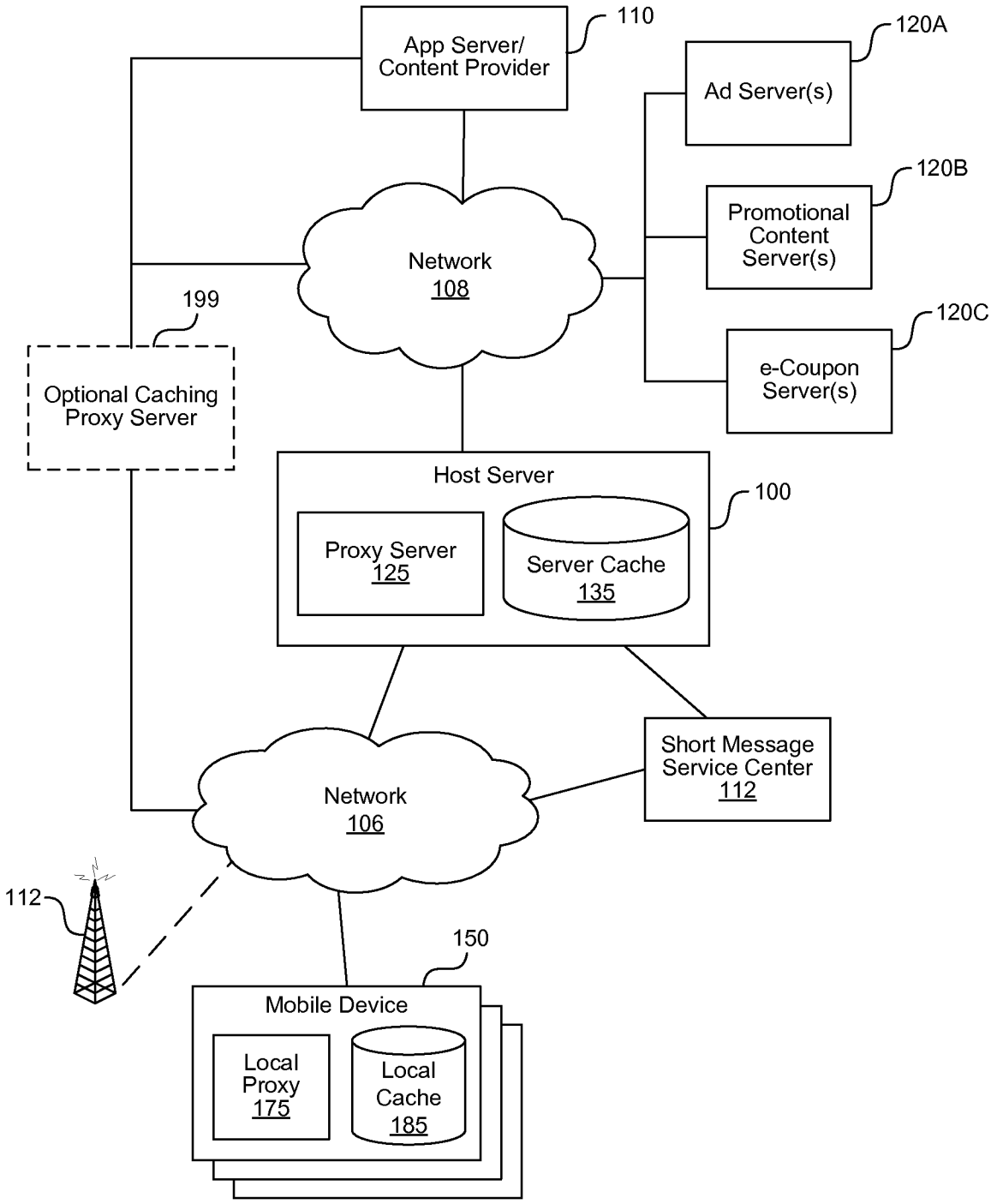

App Server/
Content Provider
110
120A
Ad Server(s)
120B
Promotional
Content
Server(s)
Network
108
199
120C
e-Coupon
Server(s)
Optional Caching
Proxy Server
Host Server
100
Proxy Server
125
Server Cache
135
Short Message
Service Center
112
Network
106
112
150
Mobile Device
Local
Proxy
175
Local
Cache
185

FIG. 1C

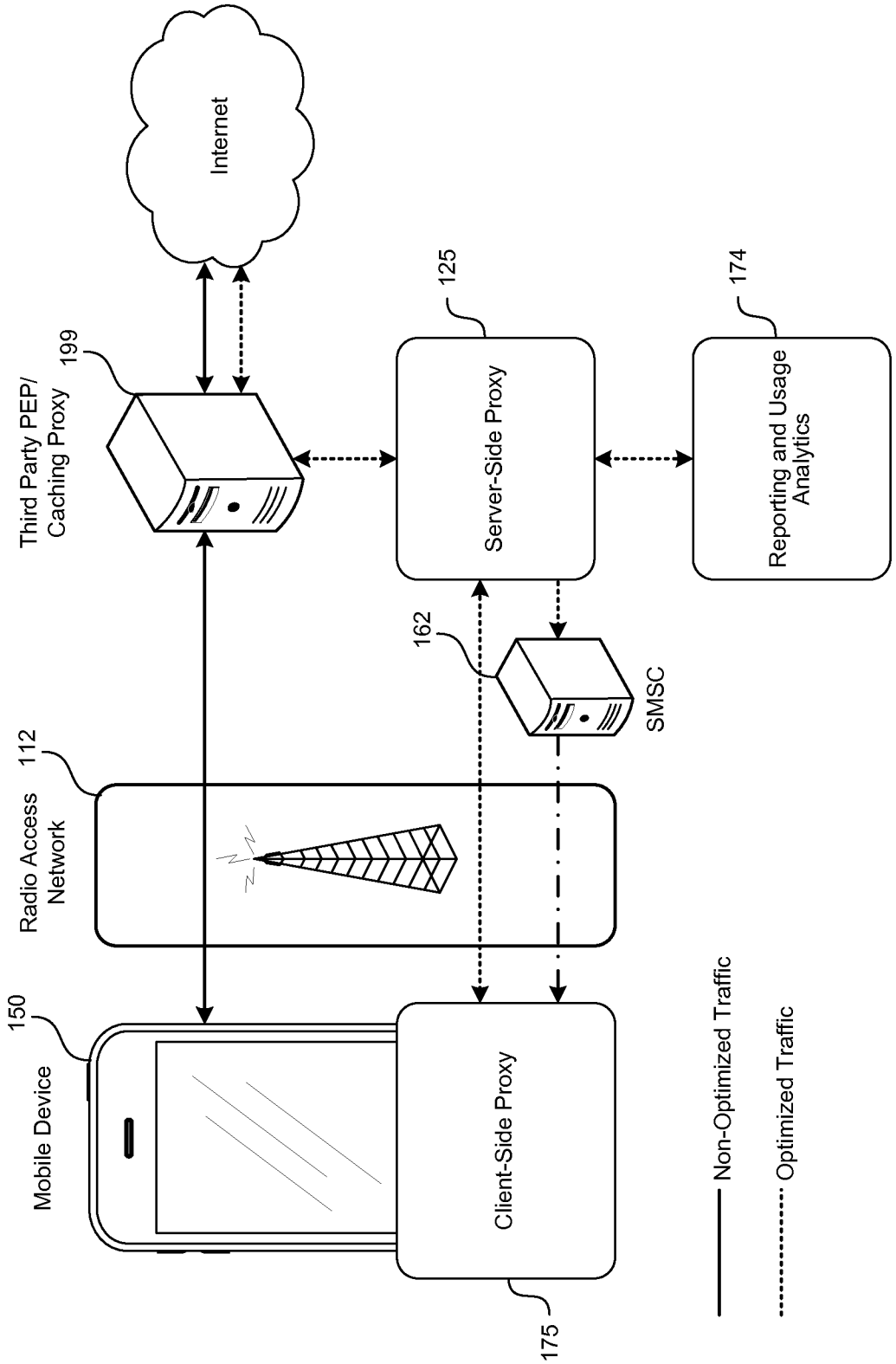

Internet
Third Party PEP/ Caching Proxy
199
Server-Side Proxy
125
Reporting and Usage Analytics
174
162
SMSC
Radio Access Network
112
Mobile Device
150
Client-Side Proxy
175
Non-Optimized Traffic
Optimized Traffic

FIG. 1D

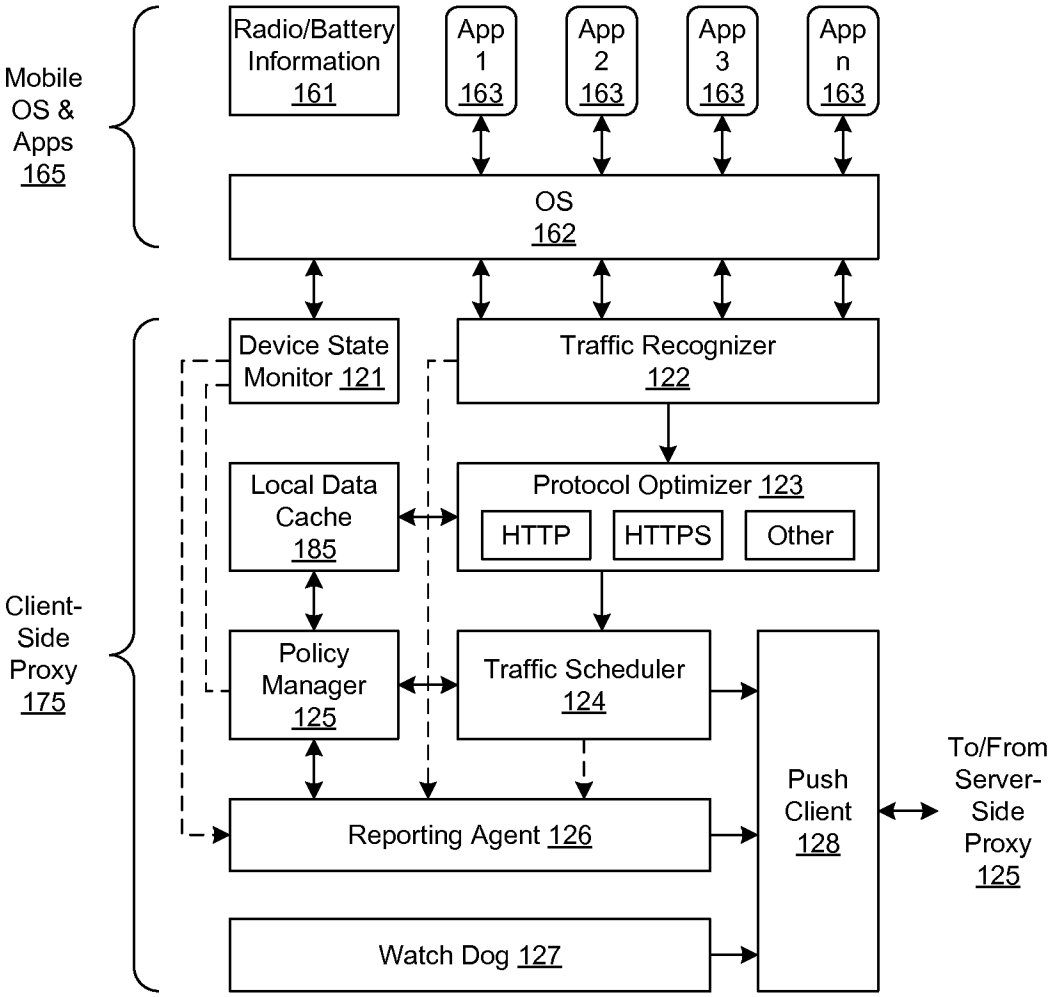

Mobile OS & Apps 165
Radio/Battery Information 161
App 1 163
App 2 163
App 3 163
App n 163
OS 162
Client-Side Proxy 175
Device State Monitor 121
Traffic Recognizer 122
Local Data Cache 185
Protocol Optimizer 123
HTTP
HTTPS
Other
Policy Manager 125
Traffic Scheduler 124
Reporting Agent 126
Watch Dog 127
Push Client 128
To/From Server-Side Proxy 125

FIG. 1E

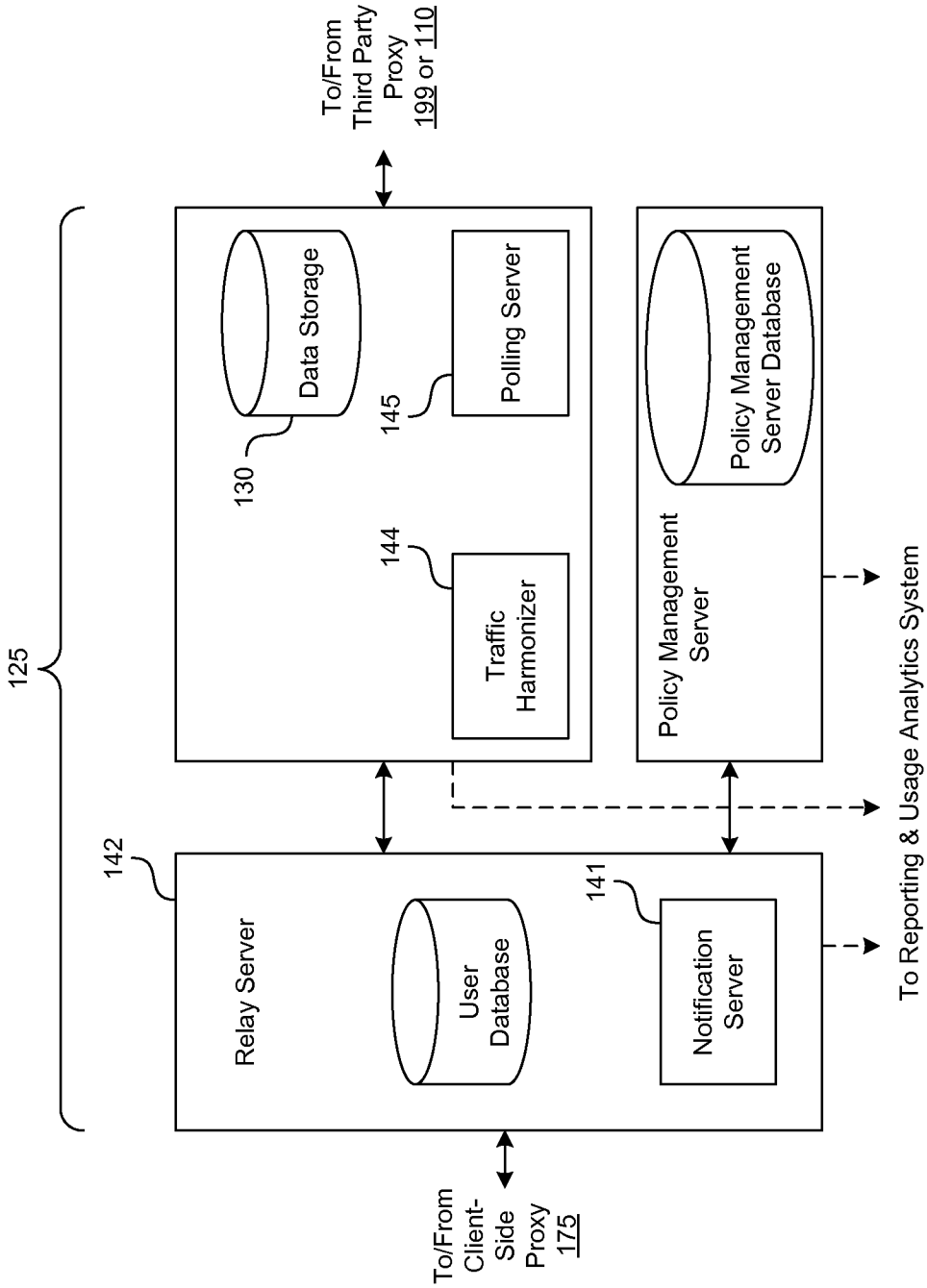

To/From Third Party Proxy 199 or 110
125
142
Relay Server
User Database
141
Notification Server
To/From Client-Side Proxy 175
Data Storage
130
145
Polling Server
144
Traffic Harmonizer
Policy Management Server
Policy Management Server Database
To Reporting & Usage Analytics System

FIG. 1F

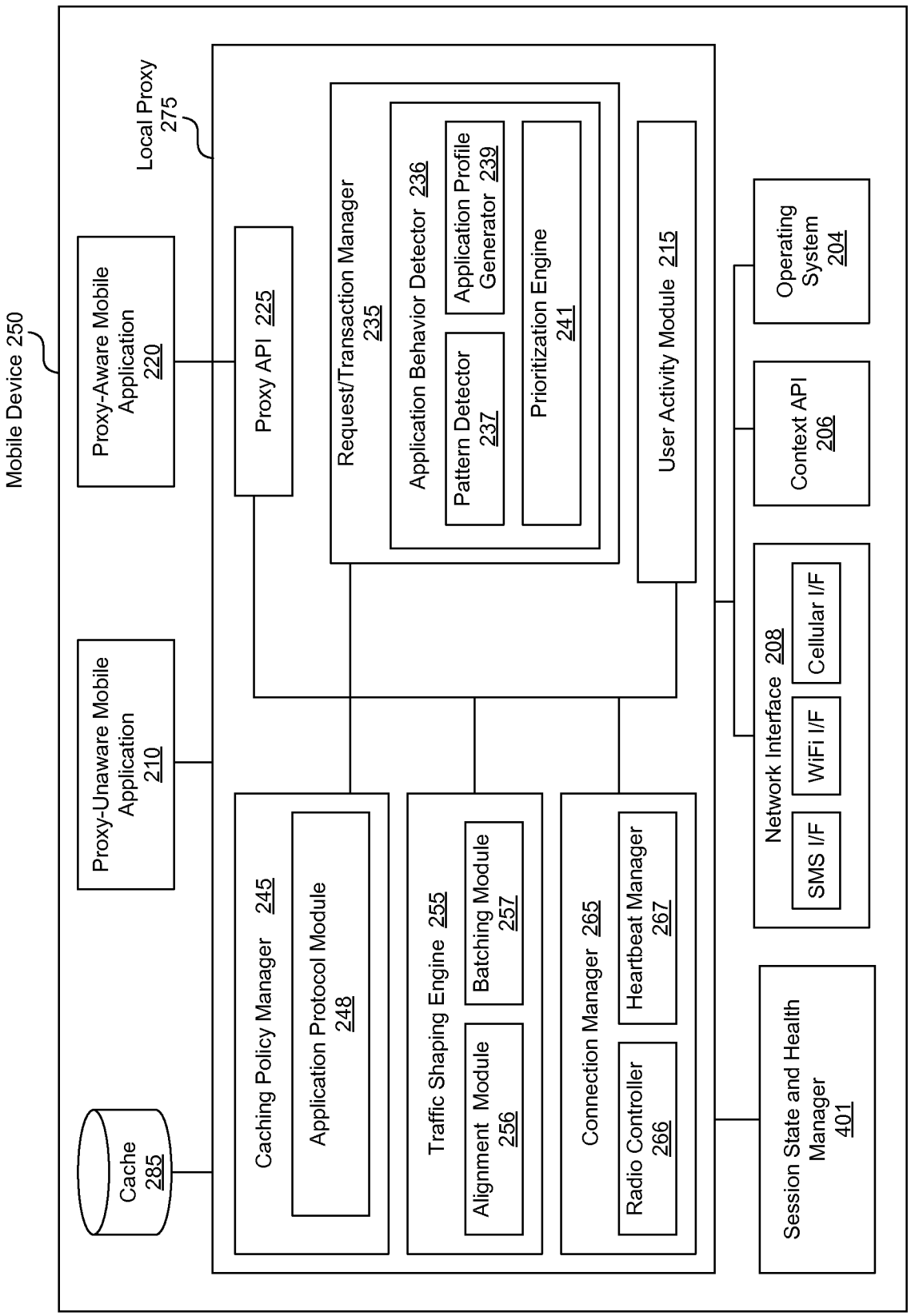

Mobile Device 250
Local Proxy 275
Proxy-Aware Mobile Application 220
Proxy-Unaware Mobile Application 210
Proxy API 225
Request/Transaction Manager 235
Application Behavior Detector 236
Pattern Detector 237
Application Profile Generator 239
Prioritization Engine 241
User Activity Module 215
Caching Policy Manager 245
Application Protocol Module 248
Traffic Shaping Engine 255
Alignment Module 256
Batching Module 257
Connection Manager 265
Radio Controller 266
Heartbeat Manager 267
Cache 285
Network Interface 208
SMS I/F
WiFi I/F
Cellular I/F
Context API 206
Operating System 204
Session State and Health Manager 401

FIG. 2A

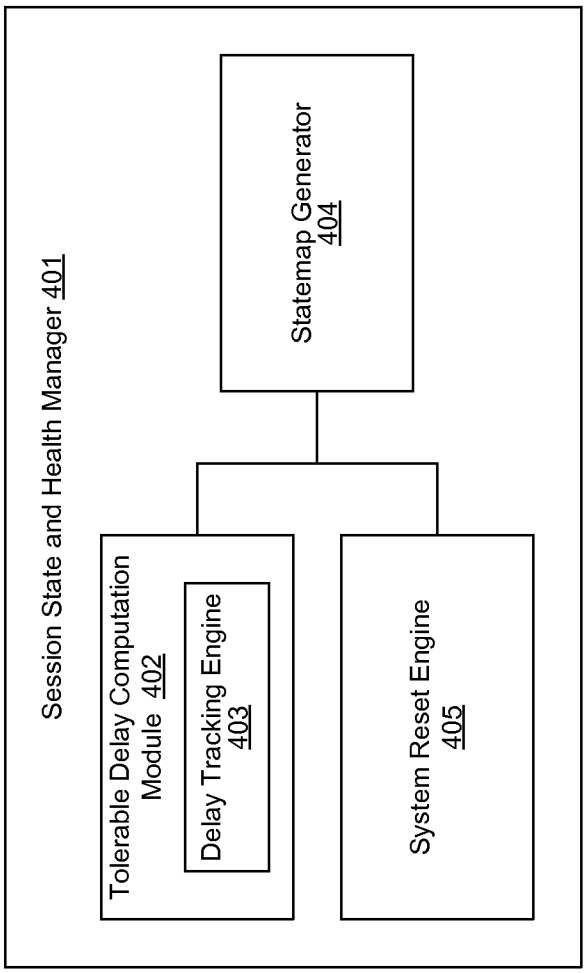

Session State and Health Manager 401
Statemap Generator 404
Tolerable Delay Computation Module 402
Delay Tracking Engine 403
System Reset Engine 405

FIG. 2B

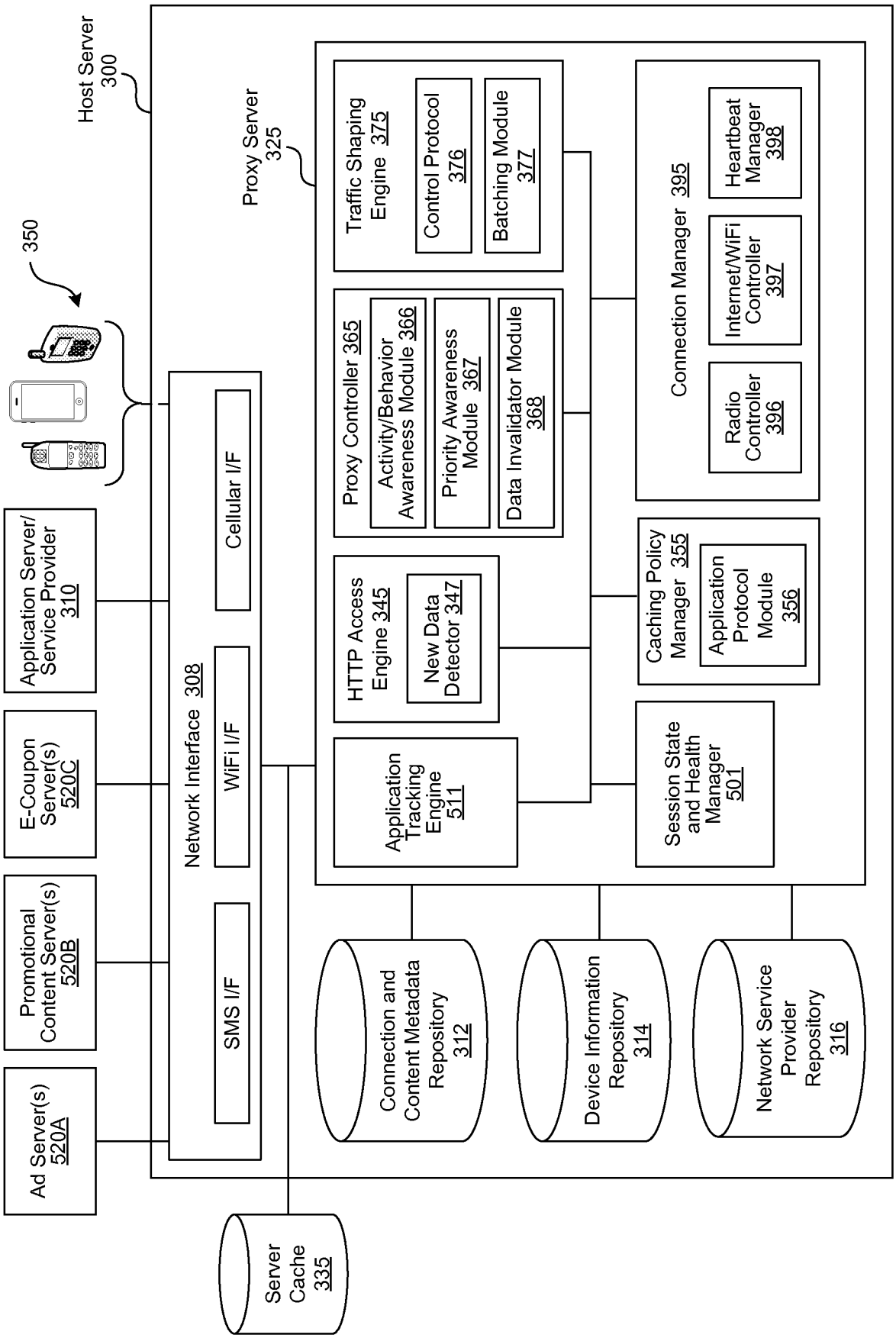

Host Server 300
Proxy Server 325
350
Application Server/ Service Provider 310
E-Coupon Server(s) 520C
Promotional Content Server(s) 520B
Ad Server(s) 520A
Network Interface 308
SMS I/F
WiFi I/F
Cellular I/F
Server Cache 335
Traffic Shaping Engine 375
Control Protocol 376
Batching Module 377
Proxy Controller 365
Activity/Behavior Awareness Module 366
Priority Awareness Module 367
Data Invalidator Module 368
HTTP Access Engine 345
New Data Detector 347
Application Tracking Engine 511
Session State and Health Manager 501
Caching Policy Manager 355
Application Protocol Module 356
Connection Manager 395
Radio Controller 396
Internet/WiFi Controller 397
Heartbeat Manager 398
Connection and Content Metadata Repository 312
Device Information Repository 314
Network Service Provider Repository 316

FIG. 3A

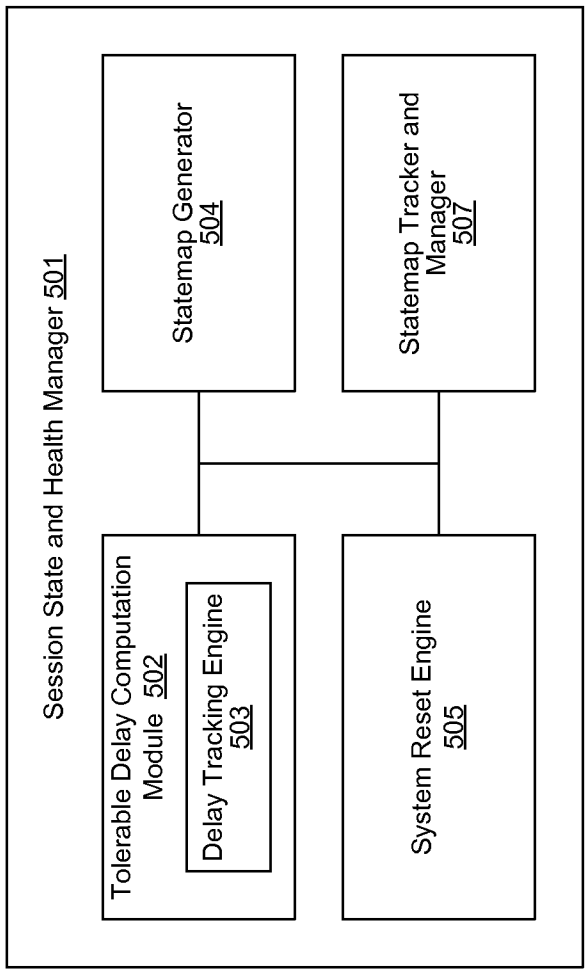

Session State and Health Manager 501
Statemap Generator 504
Statemap Tracker and Manager 507
Tolerable Delay Computation Module 502
Delay Tracking Engine 503
System Reset Engine 505

FIG. 3B

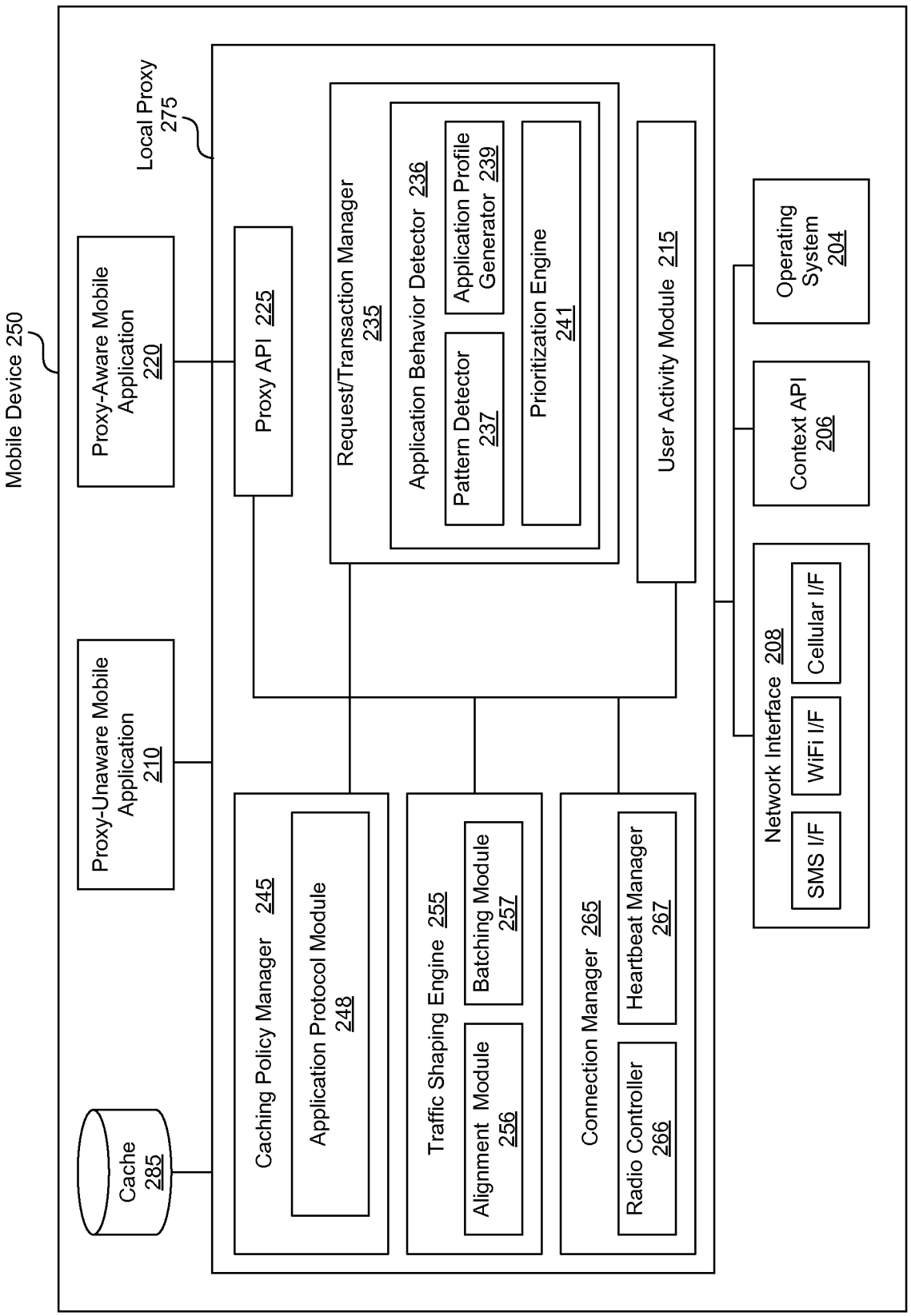

Mobile Device 250
Local Proxy 275
Proxy-Aware Mobile Application 220
Proxy-Unaware Mobile Application 210
Proxy API 225
Request/Transaction Manager 235
Application Behavior Detector 236
Pattern Detector 237
Application Profile Generator 239
Prioritization Engine 241
User Activity Module 215
Caching Policy Manager 245
Application Protocol Module 248
Traffic Shaping Engine 255
Alignment Module 256
Batching Module 257
Connection Manager 265
Radio Controller 266
Heartbeat Manager 267
Cache 285
Network Interface 208
SMS I/F
WiFi I/F
Cellular I/F
Context API 206
Operating System 204

FIG. 4A

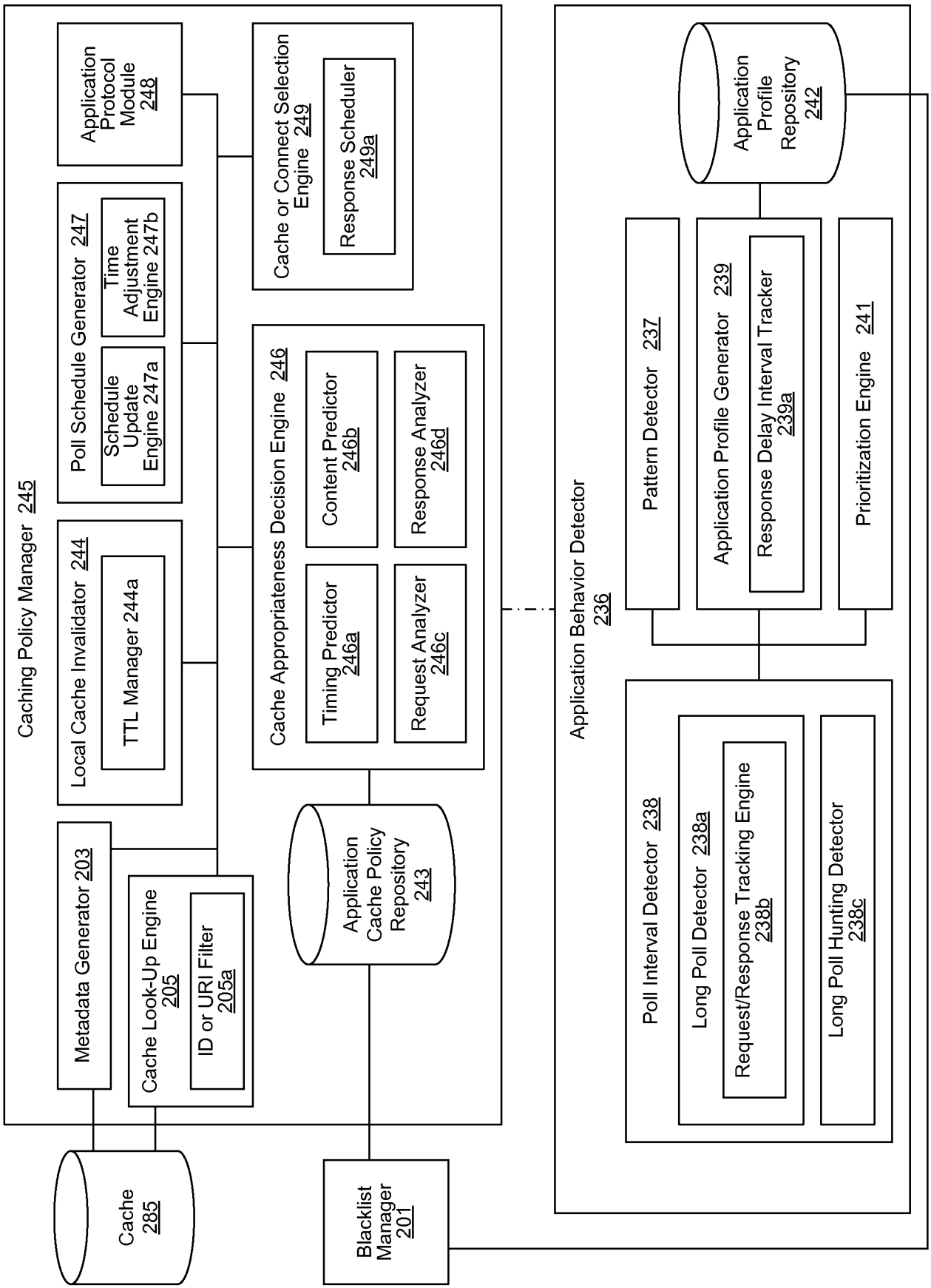

Caching Policy Manager 245
Application Protocol Module 248
Poll Schedule Generator 247
Schedule Update Engine 247a
Time Adjustment Engine 247b
Cache or Connect Selection Engine 249
Response Scheduler 249a
Local Cache Invalidator 244
TTL Manager 244a
Cache Appropriateness Decision Engine 246
Timing Predictor 246a
Content Predictor 246b
Request Analyzer 246c
Response Analyzer 246d
Metadata Generator 203
Cache Look-Up Engine 205
ID or URI Filter 205a
Application Cache Policy Repository 243
Application Behavior Detector 236
Pattern Detector 237
Application Profile Generator 239
Response Delay Interval Tracker 239a
Prioritization Engine 241
Application Profile Repository 242
Poll Interval Detector 238
Long Poll Detector 238a
Request/Response Tracking Engine 238b
Long Poll Hunting Detector 238c
Cache 285
Blacklist Manager 201

FIG. 4B

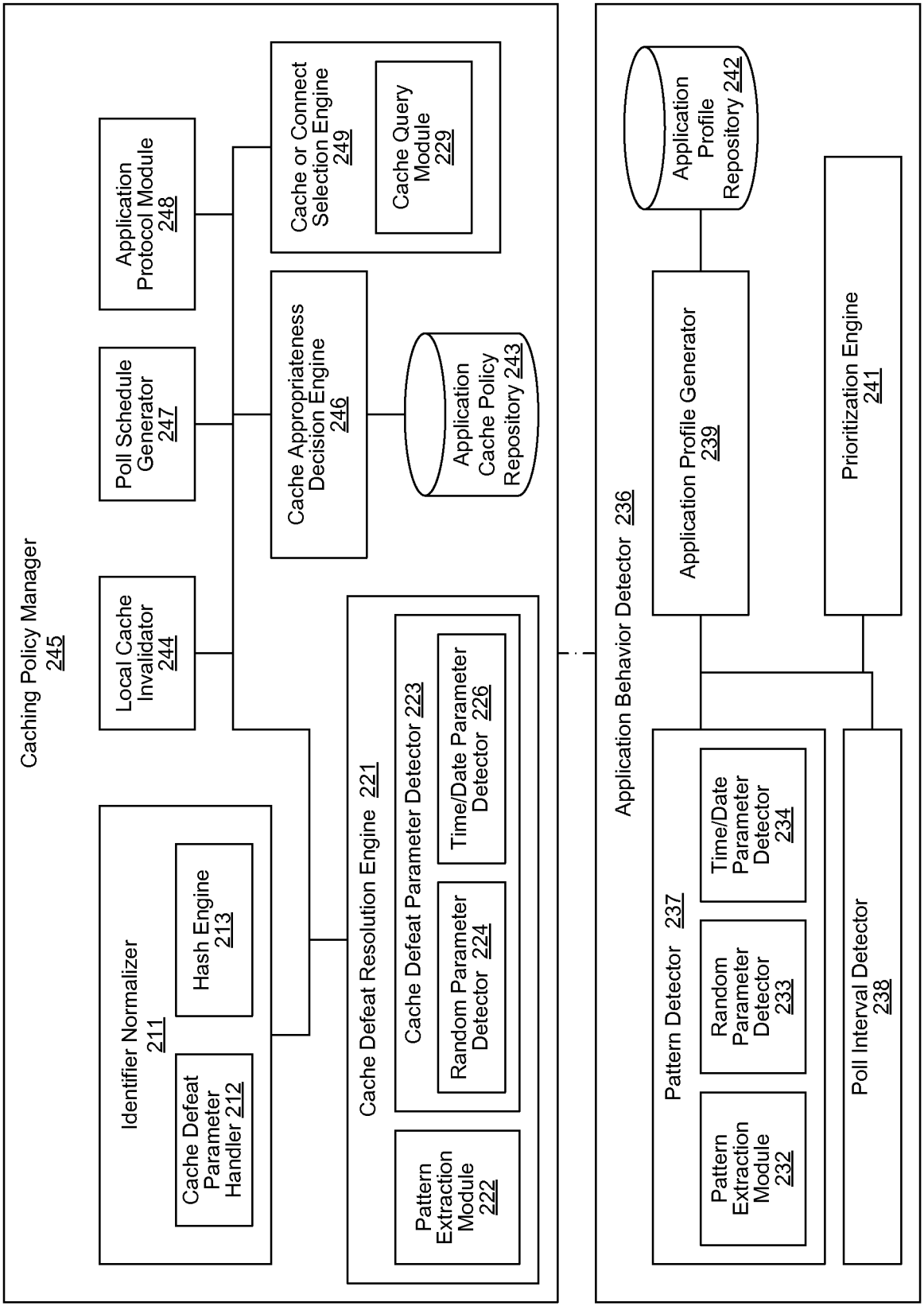

Caching Policy Manager 245
Identifier Normalizer 211
Cache Defeat Parameter Handler 212
Hash Engine 213
Local Cache Invalidator 244
Cache Defeat Resolution Engine 221
Pattern Extraction Module 222
Cache Defeat Parameter Detector 223
Random Parameter Detector 224
Time/Date Parameter Detector 226
Poll Schedule Generator 247
Application Protocol Module 248
Cache Appropriateness Decision Engine 246
Cache or Connect Selection Engine 249
Cache Query Module 229
Application Cache Policy Repository 243
Application Behavior Detector 236
Application Profile Generator 239
Application Profile Repository 242
Prioritization Engine 241
Pattern Detector 237
Pattern Extraction Module 232
Random Parameter Detector 233
Time/Date Parameter Detector 234
Poll Interval Detector 238

FIG. 4C

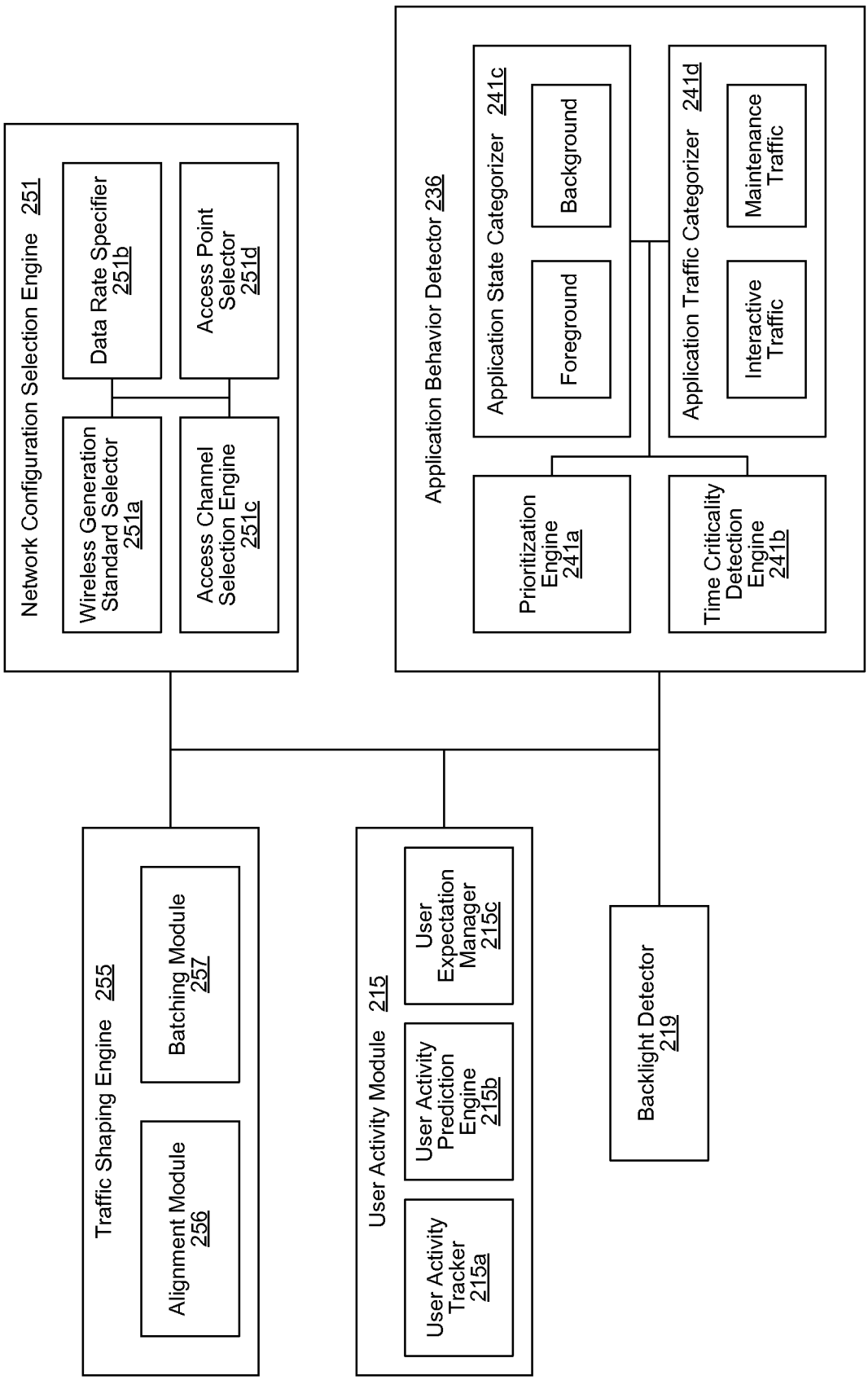

Network Configuration Selection Engine 251
Wireless Generation Standard Selector 251a
Data Rate Specifier 251b
Access Channel Selection Engine 251c
Access Point Selector 251d
Application Behavior Detector 236
Application State Categorizer 241c
Foreground
Background
Application Traffic Categorizer 241d
Interactive Traffic
Maintenance Traffic
Prioritization Engine 241a
Time Criticality Detection Engine 241b
Traffic Shaping Engine 255
Alignment Module 256
Batching Module 257
User Activity Module 215
User Activity Tracker 215a
User Activity Prediction Engine 215b
User Expectation Manager 215c
Backlight Detector 219

FIG. 4D

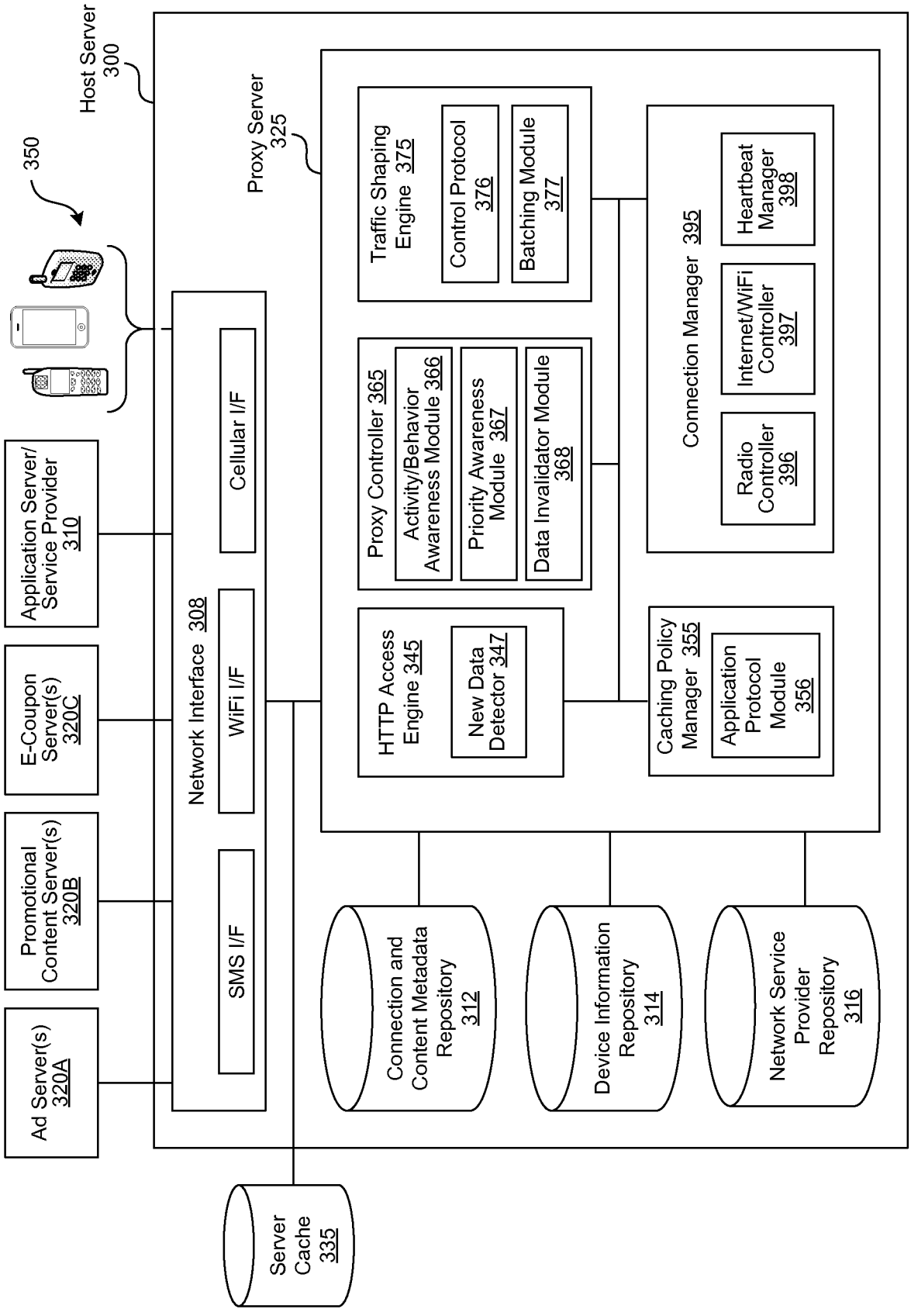

Host Server 300
350
Proxy Server 325
Application Server/ Service Provider 310
E-Coupon Server(s) 320C
Promotional Content Server(s) 320B
Ad Server(s) 320A
Network Interface 308
SMS I/F
WiFi I/F
Cellular I/F
Traffic Shaping Engine 375
Control Protocol 376
Batching Module 377
Proxy Controller 365
Activity/Behavior Awareness Module 366
Priority Awareness Module 367
Data Invalidator Module 368
Connection Manager 395
Heartbeat Manager 398
Internet/WiFi Controller 397
Radio Controller 396
HTTP Access Engine 345
New Data Detector 347
Caching Policy Manager 355
Application Protocol Module 356
Connection and Content Metadata Repository 312
Device Information Repository 314
Network Service Provider Repository 316
Server Cache 335

FIG. 5A

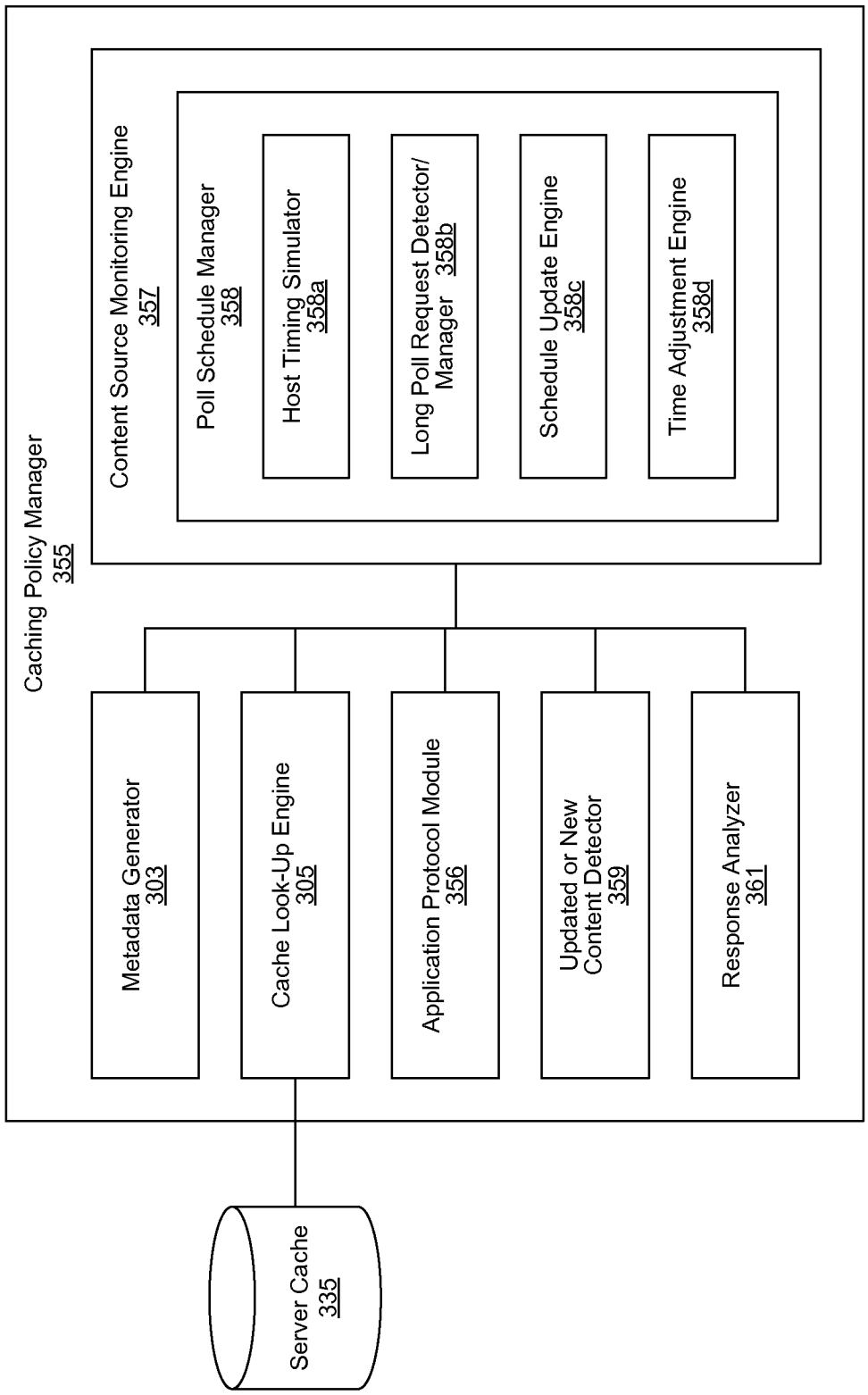

Caching Policy Manager
355
Content Source Monitoring Engine
357
Poll Schedule Manager
358
Host Timing Simulator
358a
Long Poll Request Detector/
Manager 358b
Schedule Update Engine
358c
Time Adjustment Engine
358d
Metadata Generator
303
Cache Look-Up Engine
305
Application Protocol Module
356
Updated or New
Content Detector
359
Response Analyzer
361
Server Cache
335

FIG. 5B

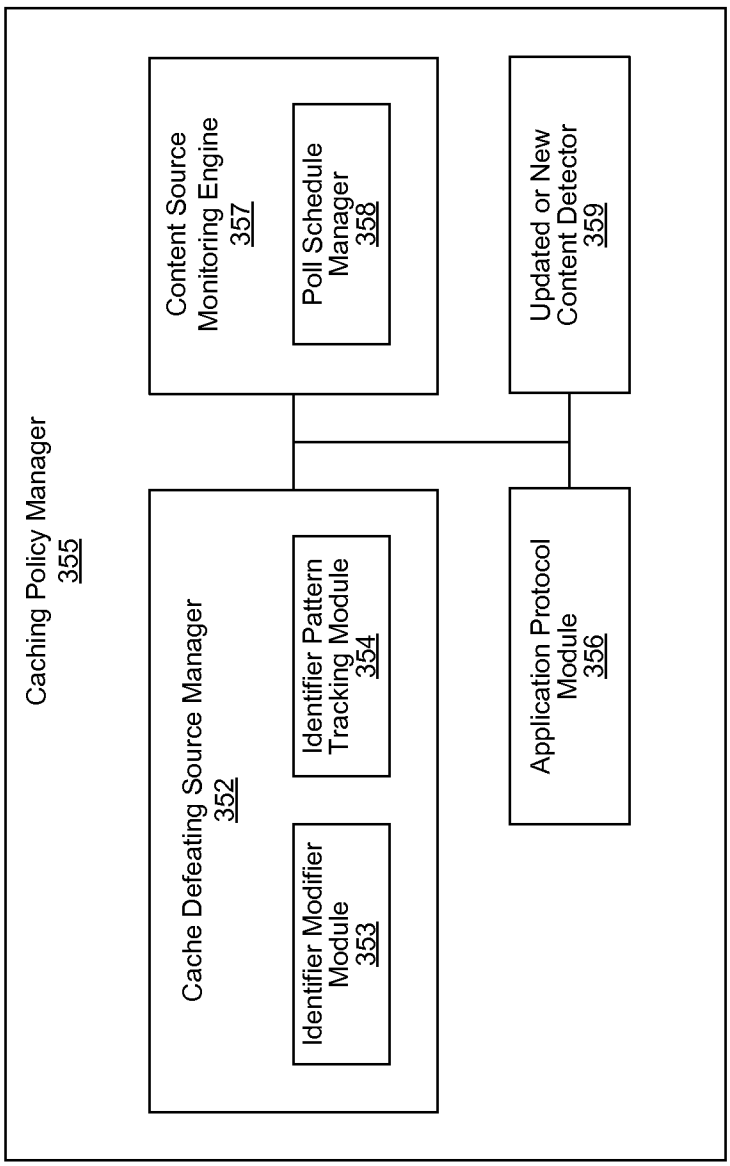

Caching Policy Manager
355
Content Source Monitoring Engine
357
Poll Schedule Manager
358
Updated or New Content Detector
359
Cache Defeating Source Manager
352
Identifier Pattern Tracking Module
354
Identifier Modifier Module
353
Application Protocol Module
356

FIG. 5C

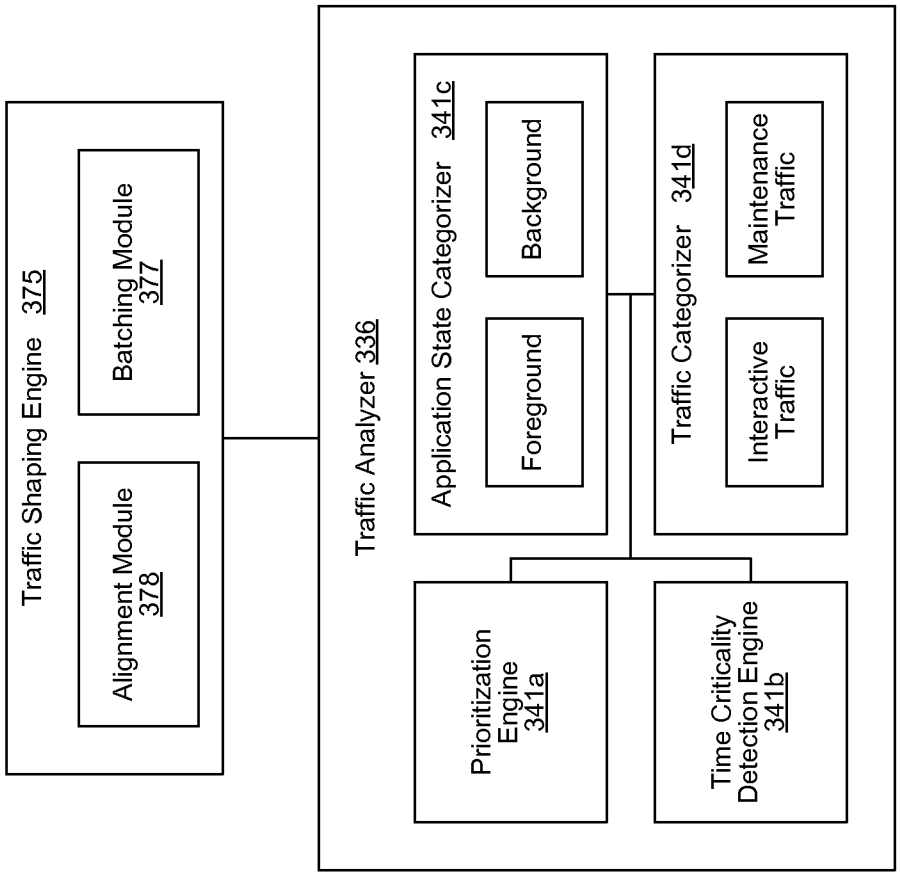

Traffic Shaping Engine 375
Alignment Module 378
Batching Module 377
Traffic Analyzer 336
Application State Categorizer 341c
Foreground
Background
Traffic Categorizer 341d
Interactive Traffic
Maintenance Traffic
Prioritization Engine 341a
Time Criticality Detection Engine 341b

FIG. 5D

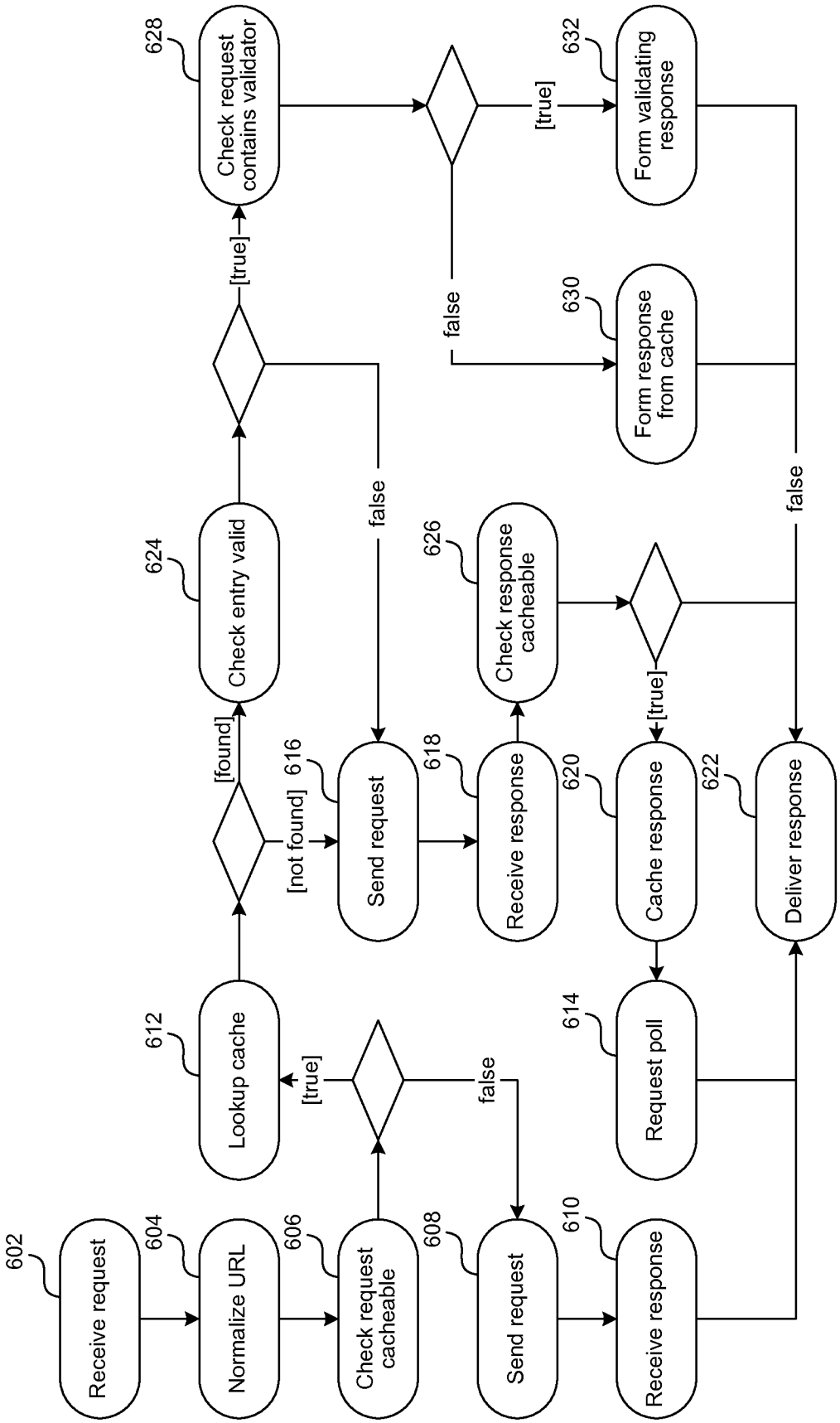

602
Receive request
604
Normalize URL
606
Check request cacheable
608
Send request
610
Receive response
612
Lookup cache
614
Request poll
616
Send request
618
Receive response
620
Cache response
622
Deliver response
624
Check entry valid
626
Check response cacheable
628
Check request contains validator
630
Form response from cache
632
Form validating response
[true]
[false]
[found]
[not found]
[true]
[false]
[true]
[false]
[true]
[false]

FIG. 6A

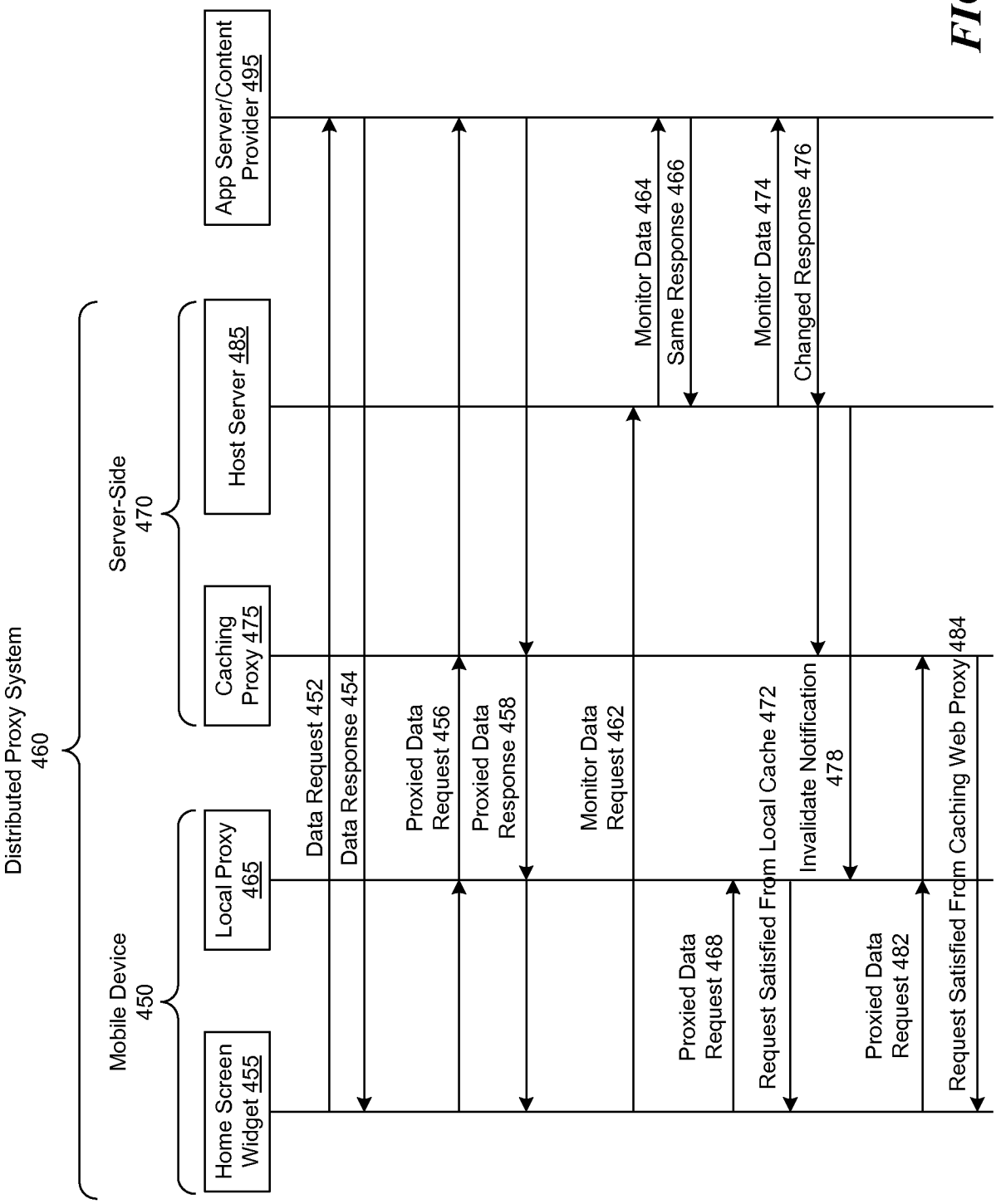

Distributed Proxy System 460
Mobile Device 450
Server-Side 470
Home Screen Widget 455
Local Proxy 465
Caching Proxy 475
Host Server 485
App Server/Content Provider 495
Data Request 452
Data Response 454
Proxied Data Request 456
Proxied Data Response 458
Monitor Data Request 462
Monitor Data 464
Same Response 466
Proxied Data Request 468
Request Satisfied From Local Cache 472
Monitor Data 474
Changed Response 476
Invalidate Notification 478
Proxied Data Request 482
Request Satisfied From Caching Web Proxy 484

FIG. 6B

Traffic Category/Application Category 700	
Interactive traffic	Background traffic
User waiting for response	User not waiting for response
Application in foreground	Application in background
Backlight on	Backlight off

FIG. 7

Content Category 800	
High priority	Low priority
Time critical	Non-time critical

FIG. 8

Mobile Application/Widget 955	Local Proxy 965	Host server 985 Server Cache 935 or Caching Proxy 975	Application Server/ Content Provider 995
Polls application server/provider 932	Poll intercepted 934		
	Proxy detects that cache content is available for the polled content and is valid and thus retrieves a response to satisfy the poll 936		
Receives a response to the poll from a cache entry 938			
Polls application server/provider 940	Poll intercepted 942		
	Proxy detects that cache content is unavailable and decides to setup the polled source for caching 944		
	Poll request forwarded to the source 946		Receives the poll request from the application and provides a response to satisfy the current request 948
Receives the response to satisfy the request from the application server/provider 950			
	Tracks polling frequency of the application and sets up a polling schedule for the host server 952		
	Sends the cache setup to the host server 954	Receives the cache setup including an identification of the application server/provider to be polled and a polling schedule 956	
		Polls the Application server/provider to monitor the response to the request 958	Receives poll from host server and sends the response 960
		Same response received, pulls the application based on the polling schedule 962	Receives poll from host server and sends the response 964
		Detects changed or new response; notifies the local proxy 966	
		Changed or new response stored in the server cache or the caching proxy 968	
	Receives notification that new or changed data is available; invalidates relevant cache entries 970		
Polls application server/content provider 972	Determines that no valid cache entry is available and retrieves the response from the server cache 974	Receives request for the new response and sends the response to the local proxy 976	
Request satisfied from the server cache or caching proxy 978			
Polls application server/content provider 980	Determines that no valid cache entry is available and forwards the poll to the application server/provider 982		Receives poll from and sends the response 984
Request satisfied from the application server/provider 986			

FIG. 9

Mobile Application/Widget 1055	Local Proxy 1065	Host server 1085 Server Cache 1035 or Caching Proxy 1075	Application Server/ Content Provider 1095
Polls application server/ provider 1032	Poll intercepted and proxy determines that a cache defeating mechanism is employed by the server/provider 1034		
	Proxy detects that cache content is available for the polled content and decides to retrieve a response to satisfy the poll 1036		
Receives a response to the poll from a cache entry 1038			
Polls application server/ provider 1040	Poll intercepted and proxy determines that a cache defeating mechanism is employed by the server/provider 1042		
	Proxy detects that cache content is unavailable and decides to setup the polled source for caching 1044		
	Extracts a pattern of an identifier of the request and tracks polling frequency of the application and sets up a polling schedule for the host server 1046		
	Poll request forwarded to the source 1048		Receives the poll request from the application and provides a response to satisfy the current request 1050
Receives the response to satisfy the request from the application server/provider 1052	Cache the response and store a normalized version of the identifier in association with the received response for future identification and retrieval 1054		
	Sends the cache setup to the host server, including the identifier or a normalized version of the identifier 1056	Receives the cache setup including an identification of the application server/provider to be polled and a polling schedule 1058	
		Polls the Application server/provider to monitor the response to the request 1060	Receives poll from host server and sends the response 1062
		Same response received, pulls the application based on the polling schedule 1064	Receives poll from host server and sends the response 1066
		Detects changed or new response; notifies the local proxy 1068	
		Changed or new response stored in the server cache or the caching proxy 1070	
	Receives notification that new or changed data is available; invalidates relevant cache entries 1072		
Polls application server/content provider 1074	Determines that no valid cache entry is available and retrieves the response from the server cache 1076	Receives request for the new response and sends the response to the local proxy 1078	
Request satisfied from the server cache or caching proxy 1080			
Polls application server/content provider 1082	Determines that no valid cache entry is available and forwards the poll to the application server/provider 1084		Receives poll from and sends the response 1086
Request satisfied from the application server/provider 1088			

FIG. 10

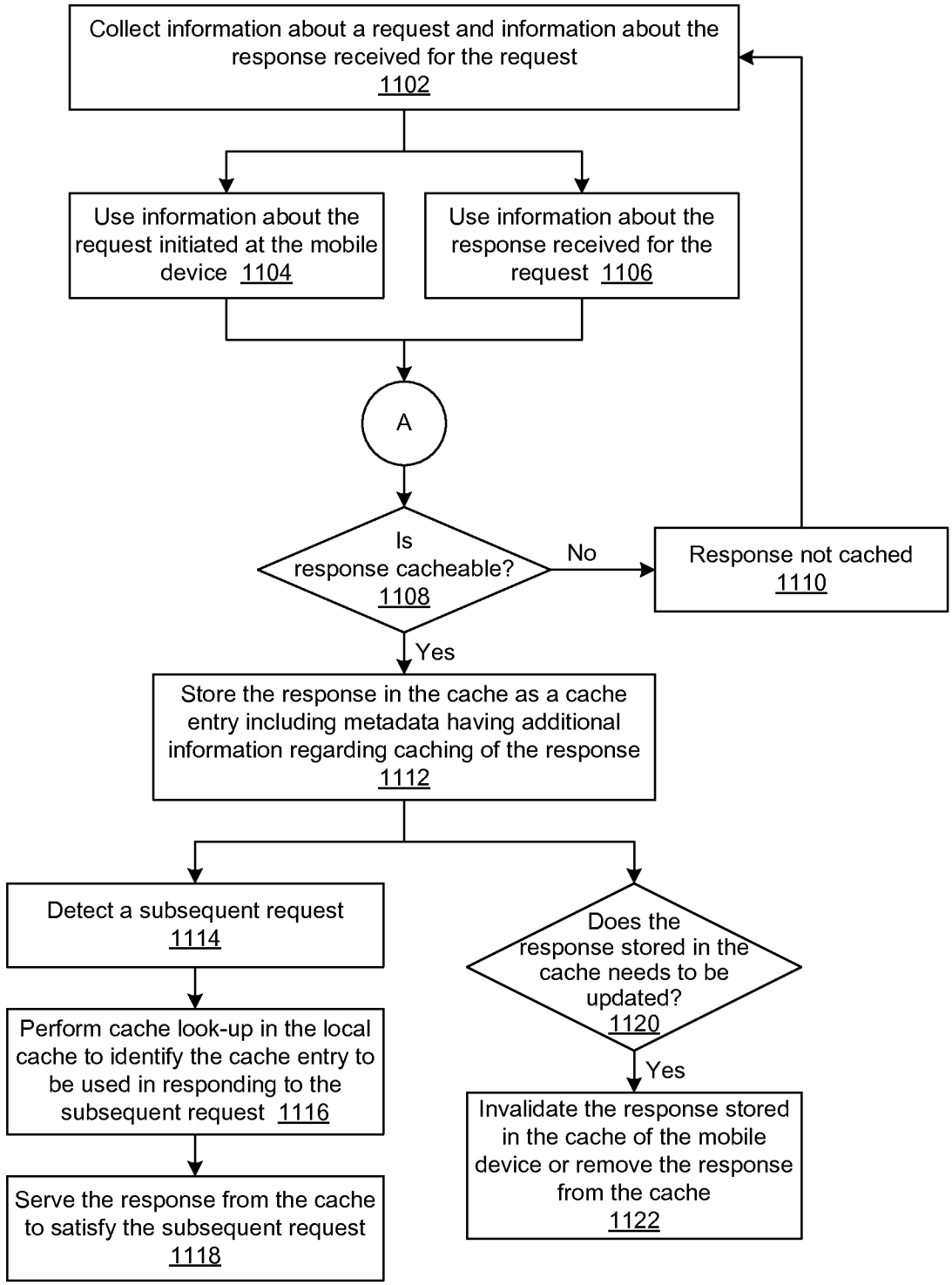

Collect information about a request and information about the response received for the request 1102
Use information about the request initiated at the mobile device 1104
Use information about the response received for the request 1106
A
Is response cacheable? 1108
No
Response not cached 1110
Yes
Store the response in the cache as a cache entry including metadata having additional information regarding caching of the response 1112
Detect a subsequent request 1114
Perform cache look-up in the local cache to identify the cache entry to be used in responding to the subsequent request 1116
Serve the response from the cache to satisfy the subsequent request 1118
Does the response stored in the cache needs to be updated? 1120
Yes
Invalidate the response stored in the cache of the mobile device or remove the response from the cache 1122

FIG. 11

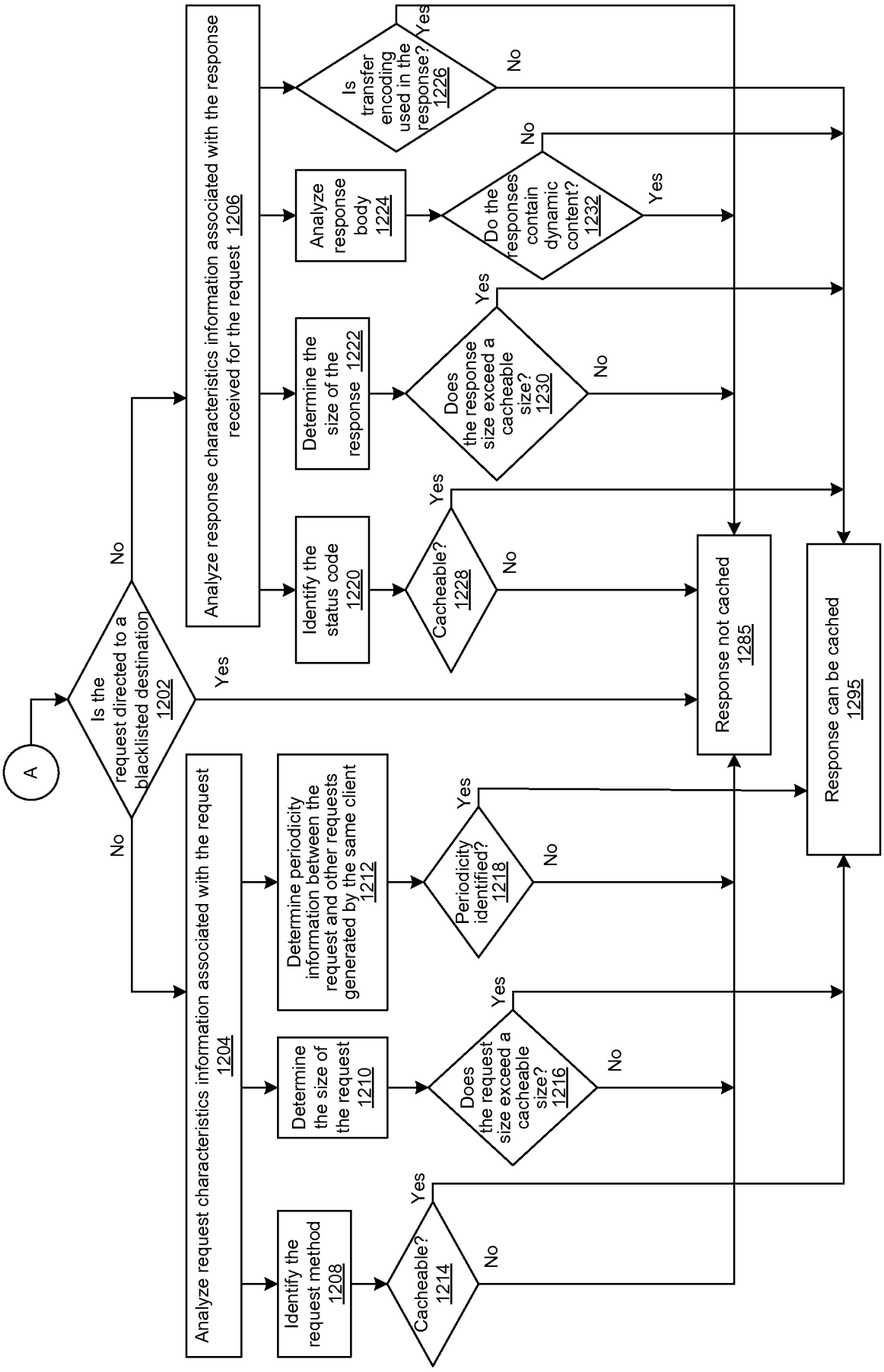

A
Is the request directed to a blacklisted destination 1202
No
Yes
Response not cached 1285
Response can be cached 1295
Analyze request characteristics information associated with the request 1204
Identify the request method 1208
Cacheable? 1214
Determine the size of the request 1210
Does the request size exceed a cacheable size? 1216
Determine periodicity information between the request and other requests generated by the same client 1212
Periodicity identified? 1218
Analyze response characteristics information associated with the response received for the request 1206
Identify the status code 1220
Cacheable? 1228
Determine the size of the response 1222
Does the response size exceed a cacheable size? 1230
Analyze response body 1224
Do the responses contain dynamic content? 1232
Is transfer encoding used in the response? 1226

FIG. 12

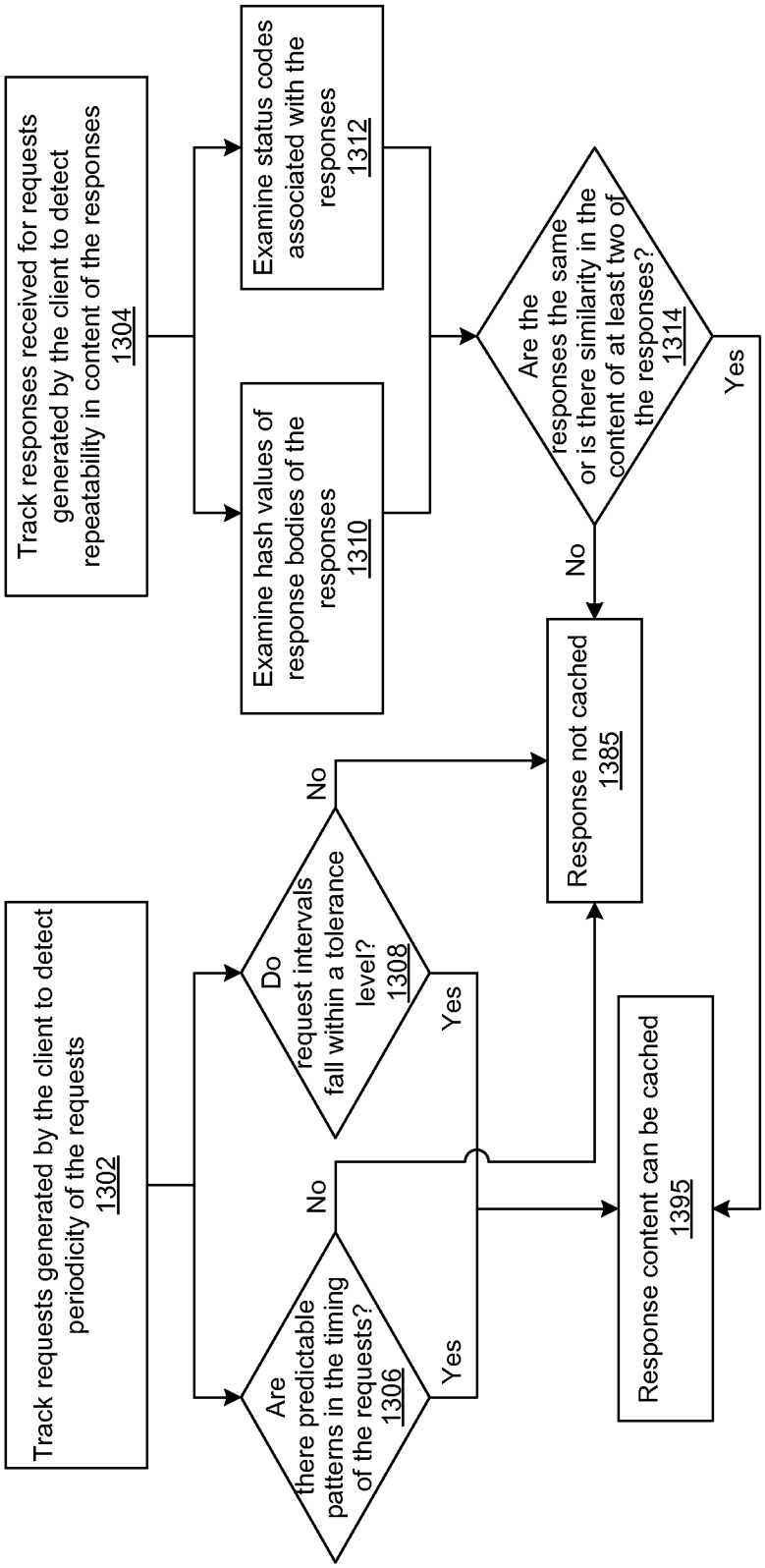

Track requests generated by the client to detect periodicity of the requests 1302
Track responses received for requests generated by the client to detect repeatability in content of the responses 1304
Are there predictable patterns in the timing of the requests? 1306
Do request intervals fall within a tolerance level? 1308
Examine hash values of response bodies of the responses 1310
Examine status codes associated with the responses 1312
Are the responses the same or is there similarity in the content of at least two of the responses? 1314
Response not cached 1385
Response content can be cached 1395
Yes
No
Yes
No
Yes
No

FIG. 13

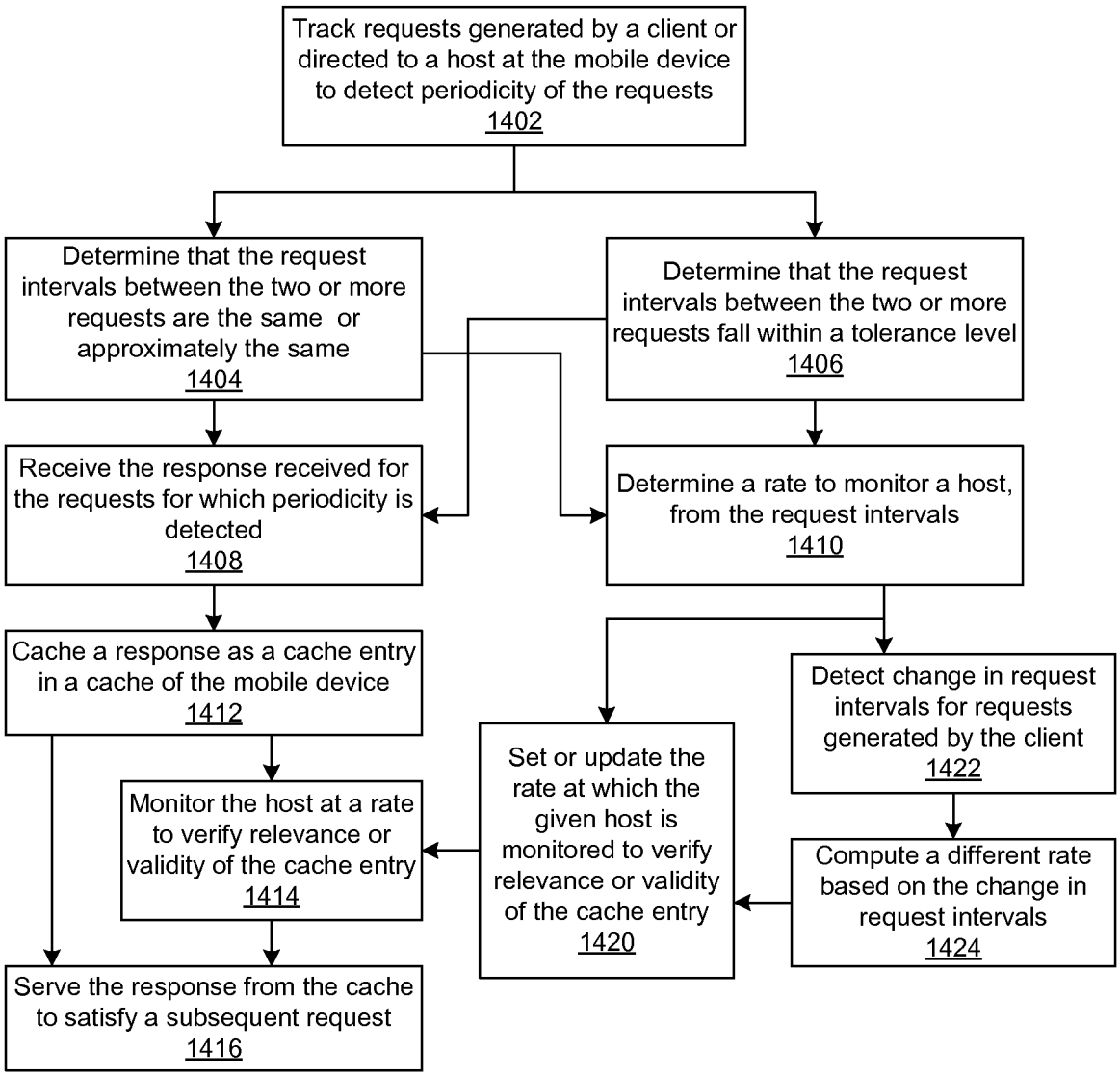

Track requests generated by a client or directed to a host at the mobile device to detect periodicity of the requests
1402
Determine that the request intervals between the two or more requests are the same or approximately the same
1404
Determine that the request intervals between the two or more requests fall within a tolerance level
1406
Receive the response received for the requests for which periodicity is detected
1408
Determine a rate to monitor a host, from the request intervals
1410
Cache a response as a cache entry in a cache of the mobile device
1412
Detect change in request intervals for requests generated by the client
1422
Monitor the host at a rate to verify relevance or validity of the cache entry
1414
Set or update the rate at which the given host is monitored to verify relevance or validity of the cache entry
1420
Compute a different rate based on the change in request intervals
1424
Serve the response from the cache to satisfy a subsequent request
1416

FIG. 14

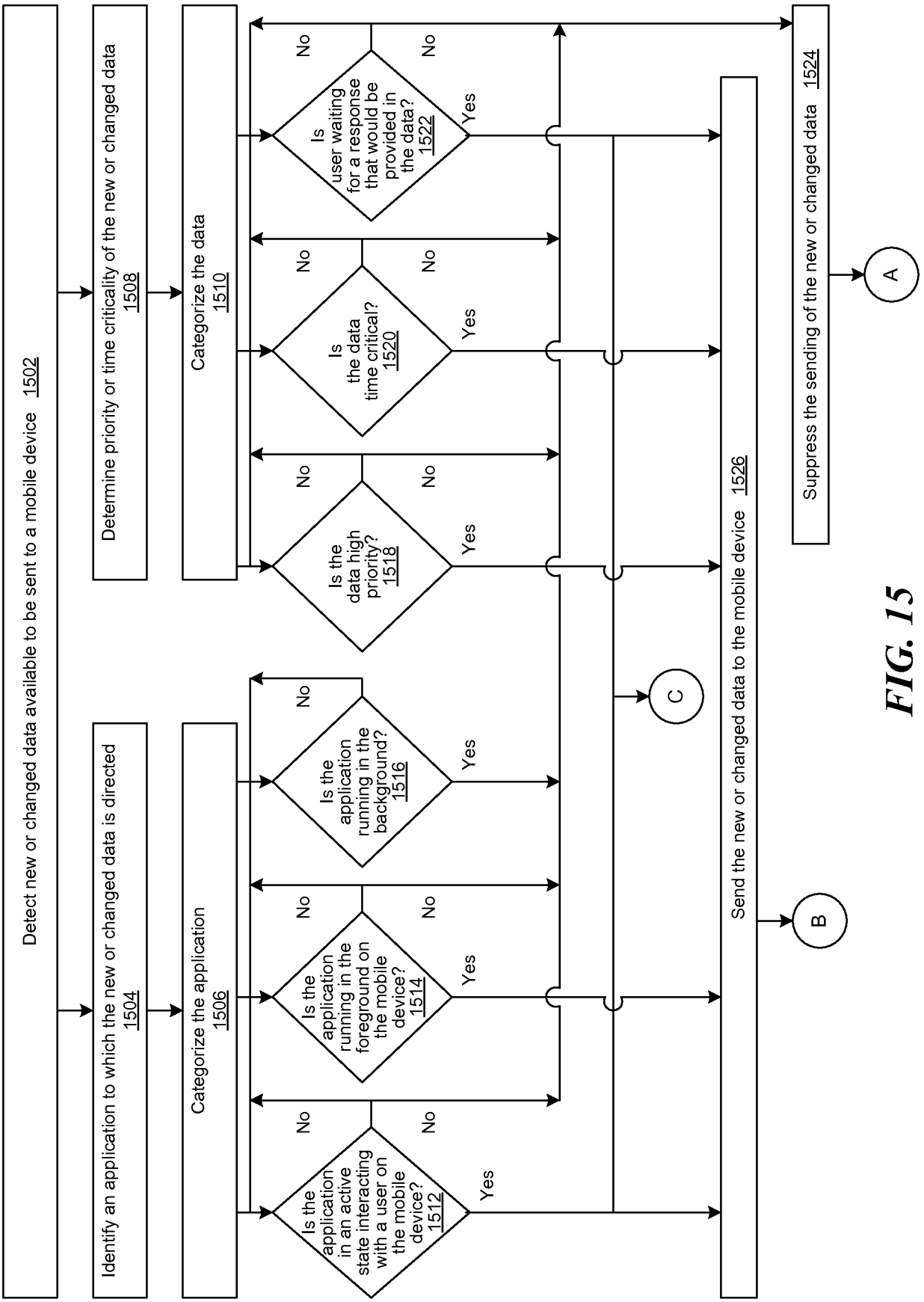

Detect new or changed data available to be sent to a mobile device 1502
Identify an application to which the new or changed data is directed 1504
Determine priority or time criticality of the new or changed data 1508
Categorize the application 1506
Categorize the data 1510
Is the application in an active state interacting with a user on the mobile device? 1512
Is the application running in the foreground on the mobile device? 1514
Is the application running in the background? 1516
Is the data high priority? 1518
Is the data time critical? 1520
Is user waiting for a response that would be provided in the data? 1522
Yes
No
A
B
C
Send the new or changed data to the mobile device 1526
Suppress the sending of the new or changed data 1524

FIG. 15

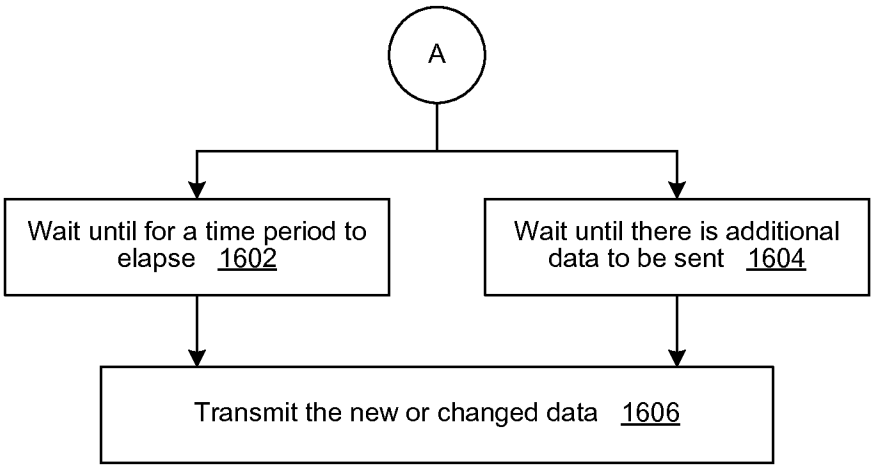

A
Wait until for a time period to elapse 1602
Wait until there is additional data to be sent 1604
Transmit the new or changed data 1606

FIG. 16A

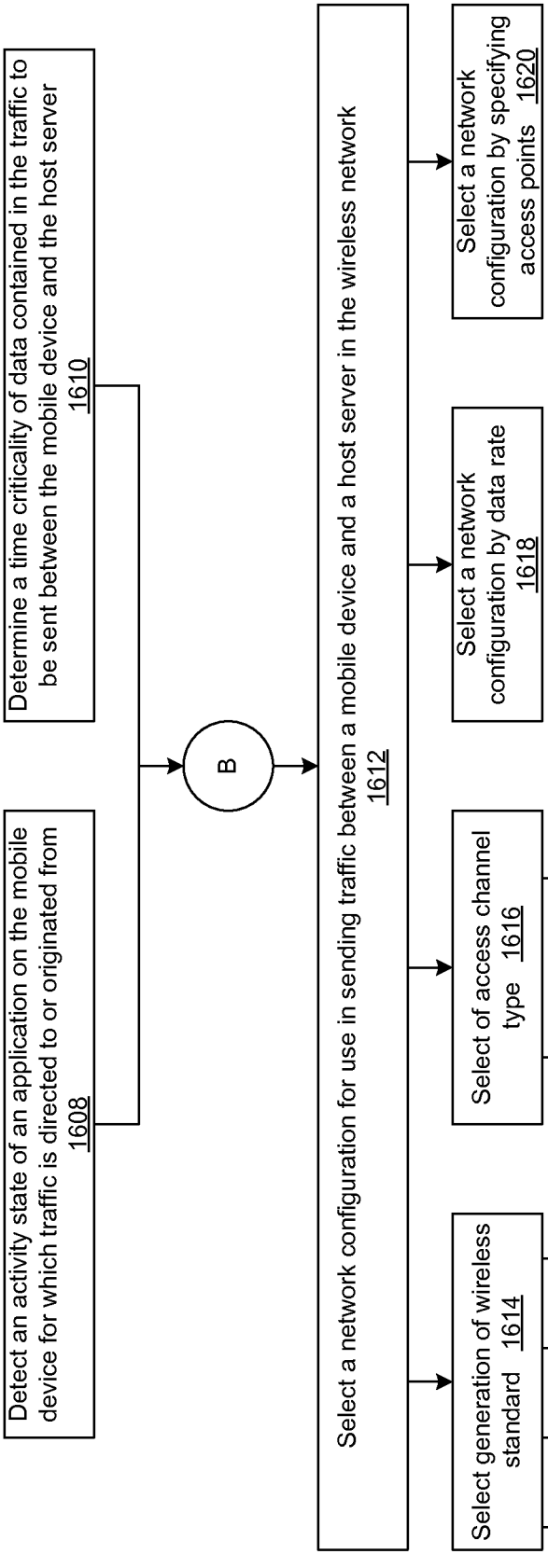

Detect an activity state of an application on the mobile device for which traffic is directed to or originated from 1608
Determine a time criticality of data contained in the traffic to be sent between the mobile device and the host server 1610
B
Select a network configuration for use in sending traffic between a mobile device and a host server in the wireless network 1612
Select generation of wireless standard 1614
2G/2.5G 1622
3G 1624
LTE 1626
4G 1628
Select of access channel type 1616
Forward access channel (FACH) 1630
Dedicated channel (DCH) 1632
Select a network configuration by data rate 1618
Select a network configuration by specifying access points 1620

FIG. 16B

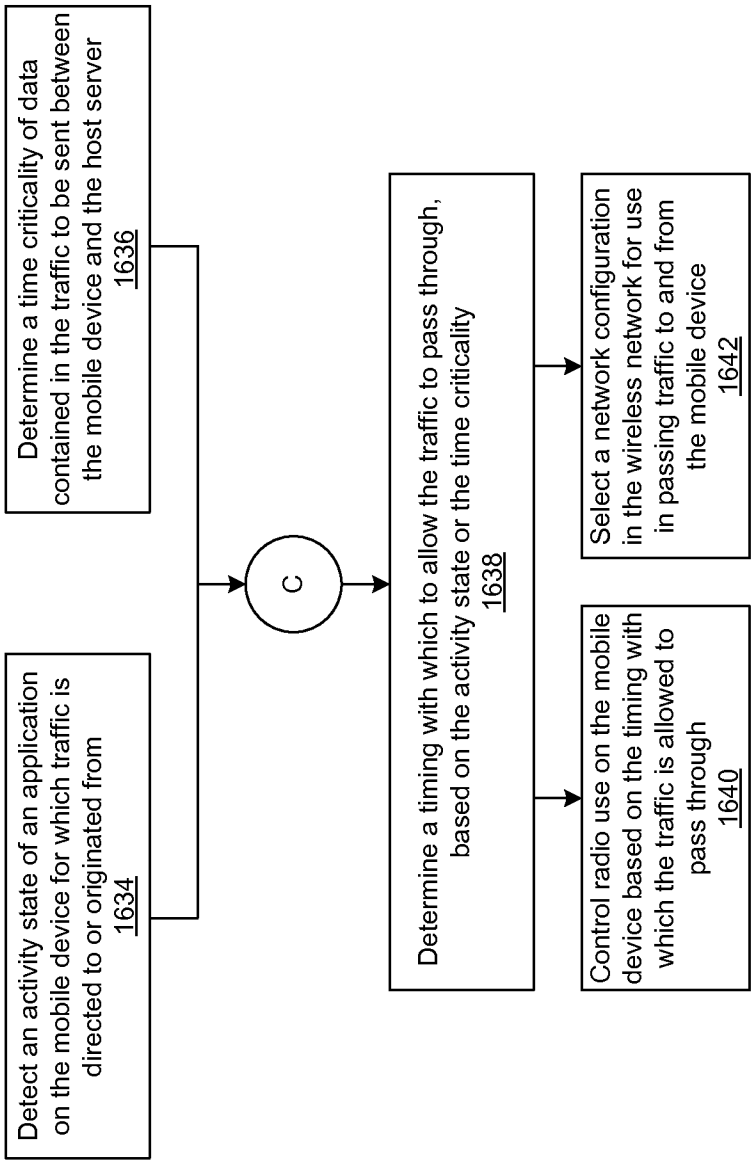

Detect an activity state of an application on the mobile device for which traffic is directed to or originated from
1634
Determine a time criticality of data contained in the traffic to be sent between the mobile device and the host server
1636
C
Determine a timing with which to allow the traffic to pass through, based on the activity state or the time criticality
1638
Control radio use on the mobile device based on the timing with which the traffic is allowed to pass through
1640
Select a network configuration in the wireless network for use in passing traffic to and from the mobile device
1642

FIG. 16C

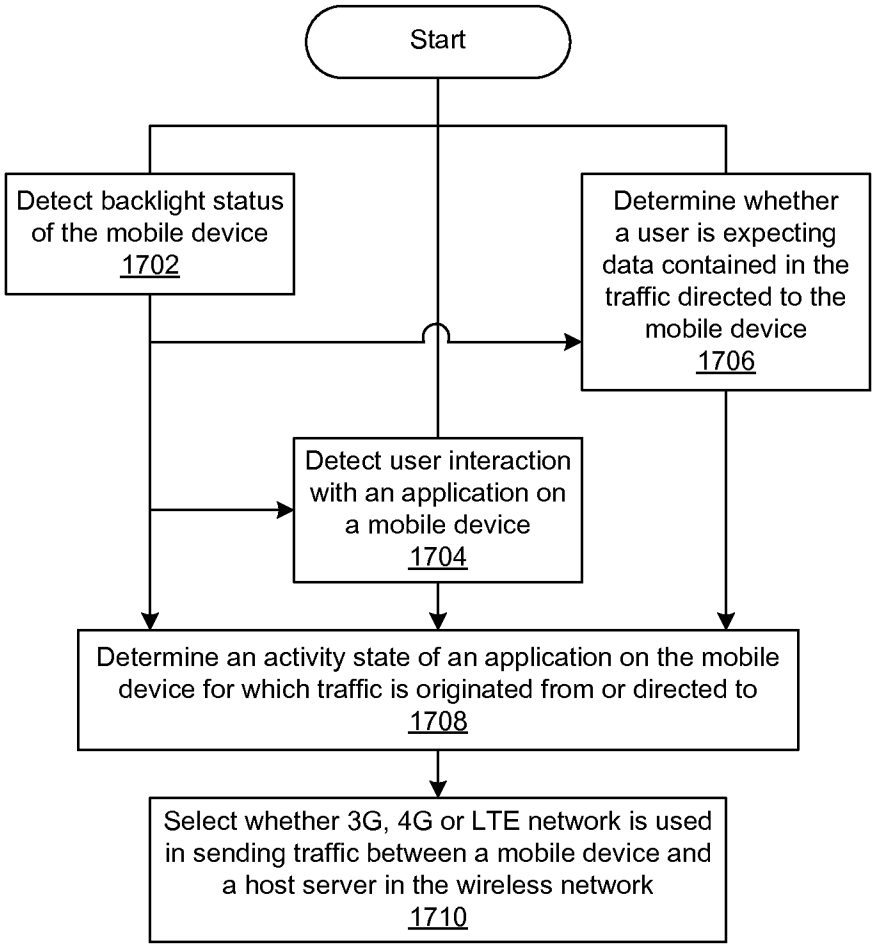

Start
Detect backlight status of the mobile device 1702
Determine whether a user is expecting data contained in the traffic directed to the mobile device 1706
Detect user interaction with an application on a mobile device 1704
Determine an activity state of an application on the mobile device for which traffic is originated from or directed to 1708
Select whether 3G, 4G or LTE network is used in sending traffic between a mobile device and a host server in the wireless network 1710

FIG. 17

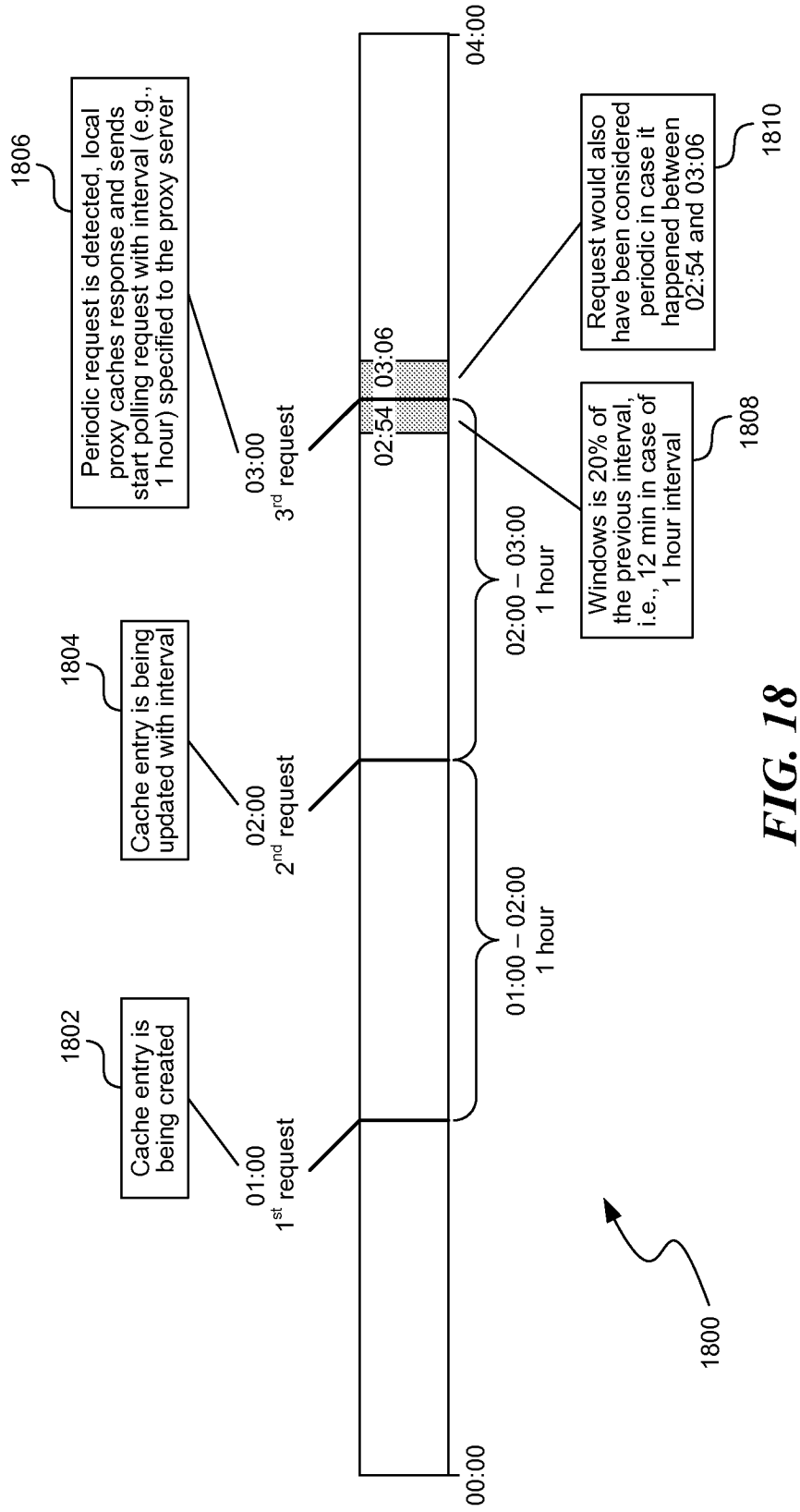

1800
00:00
04:00
01:00
1st request
02:00
2nd request
03:00
3rd request
02:54
03:06
01:00 – 02:00
1 hour
02:00 – 03:00
1 hour
1802
Cache entry is being created
1804
Cache entry is being updated with interval
1806
Periodic request is detected, local proxy caches response and sends start polling request with interval (e.g., 1 hour) specified to the proxy server
1808
Windows is 20% of the previous interval, i.e., 12 min in case of 1 hour interval
1810
Request would also have been considered periodic in case it happened between 02:54 and 03:06

FIG. 18

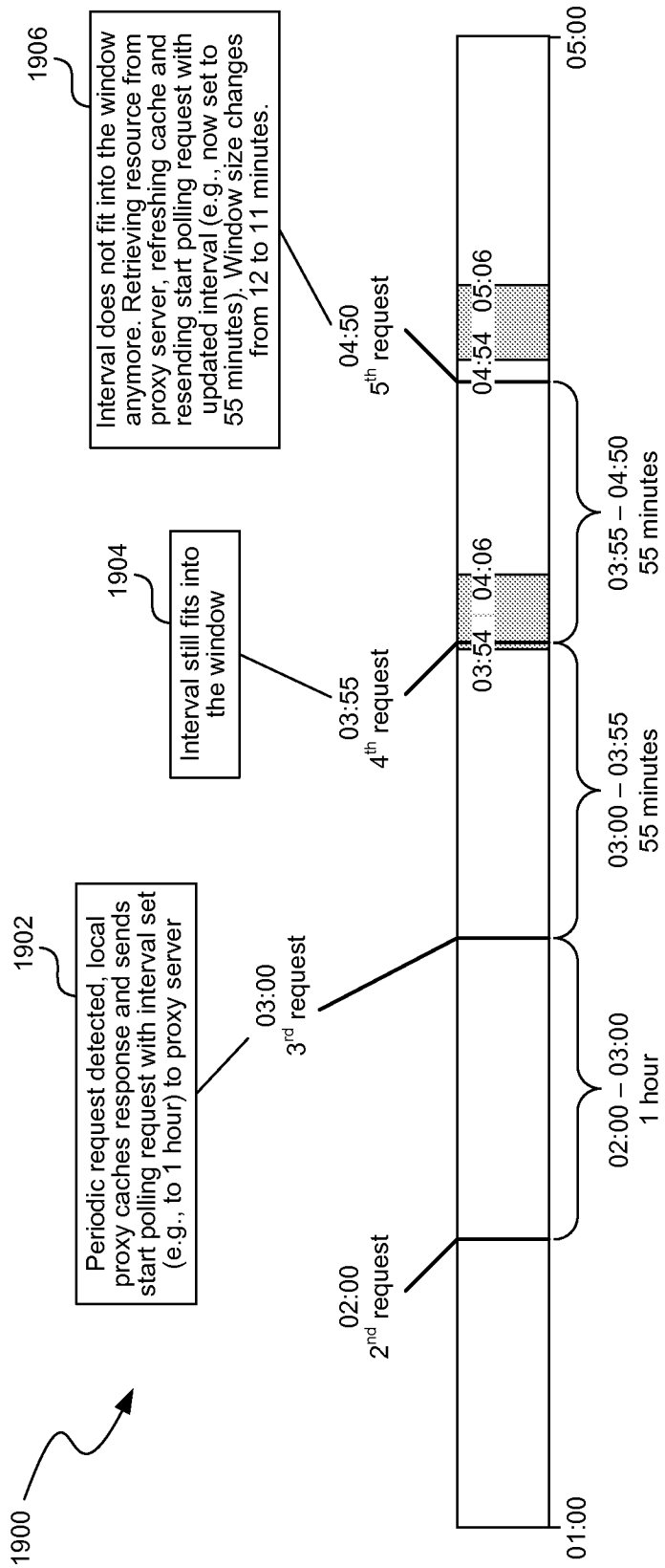

1900
1902
Periodic request detected, local proxy caches response and sends start polling request with interval set (e.g., to 1 hour) to proxy server
1904
Interval still fits into the window
1906
Interval does not fit into the window anymore. Retrieving resource from proxy server, refreshing cache and resending start polling request with updated interval (e.g., now set to 55 minutes). Window size changes from 12 to 11 minutes.
01:00
02:00
2nd request
03:00
3rd request
03:54
03:55
4th request
04:06
04:50
5th request
04:54
05:06
05:00
02:00 – 03:00
1 hour
03:00 – 03:55
55 minutes
03:55 – 04:50
55 minutes

FIG. 19

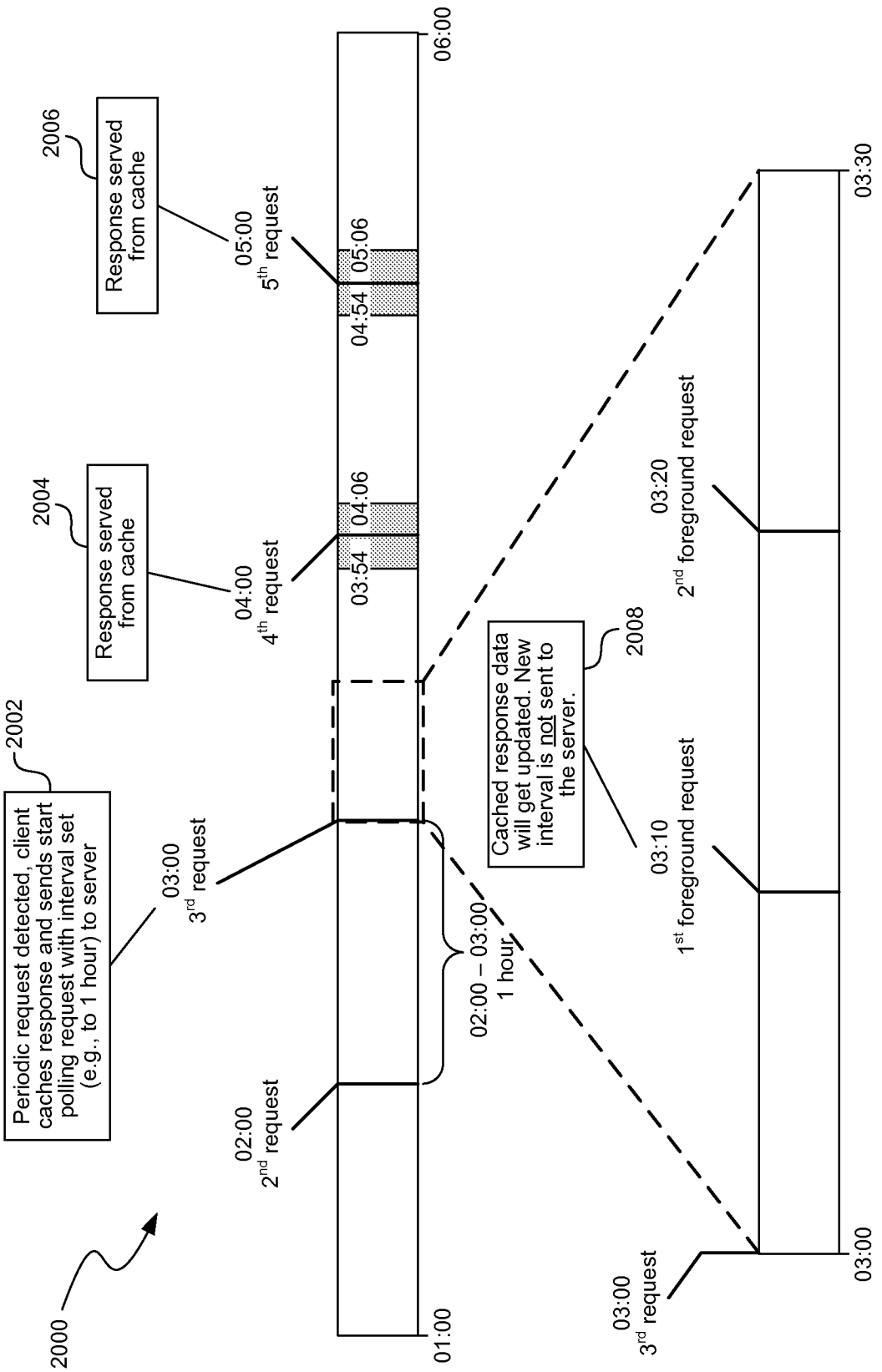

2000
Periodic request detected, client caches response and sends start polling request with interval set (e.g., to 1 hour) to server
2002
01:00
02:00
2nd request
02:00 – 03:00
1 hour
03:00
3rd request
Response served from cache
2004
04:00
4th request
03:54
04:06
Response served from cache
2006
05:00
5th request
04:54
05:06
06:00
Cached response data will get updated. New interval is not sent to the server.
2008
03:00
3rd request
03:10
1st foreground request
03:20
2nd foreground request
03:30

FIG. 20

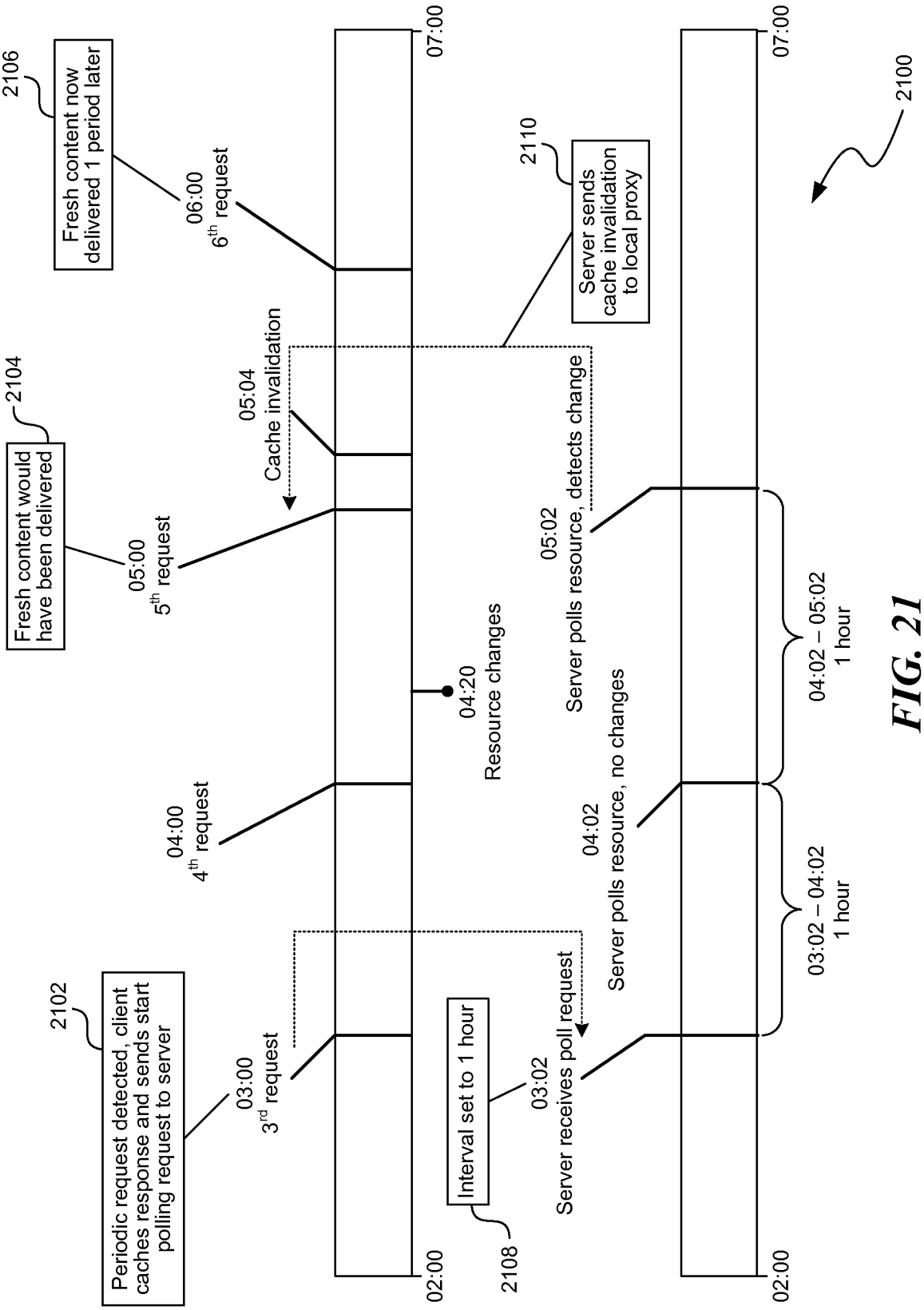

2100
2102
Periodic request detected, client caches response and sends start polling request to server
2104
Fresh content would have been delivered
2106
Fresh content now delivered 1 period later
2108
Interval set to 1 hour
2110
Server sends cache invalidation to local proxy
02:00
03:00
3rd request
04:00
4th request
05:00
5th request
06:00
6th request
07:00
03:02
Server receives poll request
04:02
Server polls resource, no changes
04:20
Resource changes
05:02
Server polls resource, detects change
05:04
Cache invalidation
03:02 – 04:02
1 hour
04:02 – 05:02
1 hour

FIG. 21

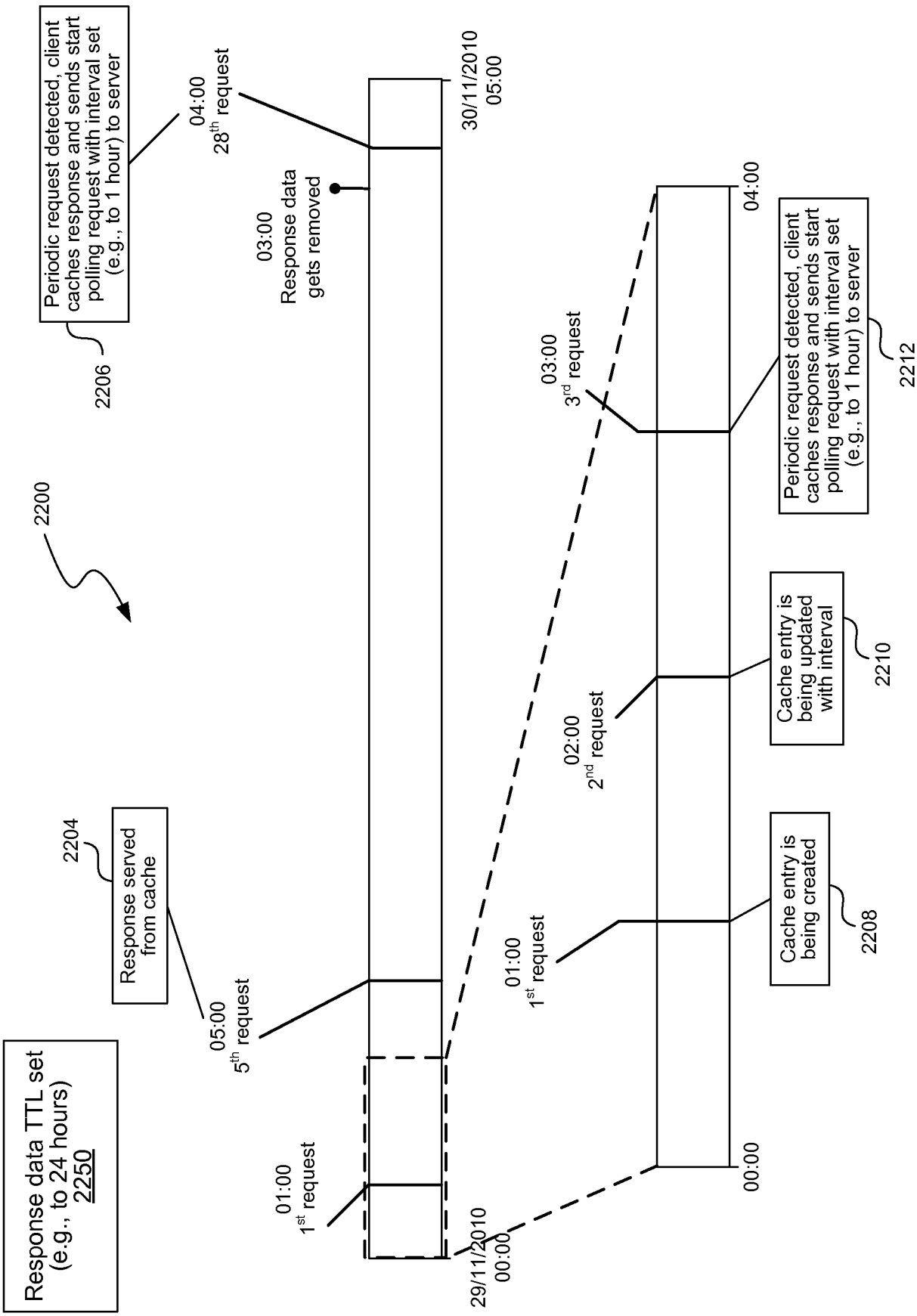

Response data TTL set (e.g., to 24 hours) 2250
2200
01:00 1st request
29/11/2010 00:00
05:00 5th request
Response served from cache
2204
01:00 1st request
Cache entry is being created
2208
02:00 2nd request
Cache entry is being updated with interval
2210
03:00 3rd request
Periodic request detected, client caches response and sends start polling request with interval set (e.g., to 1 hour) to server
2212
00:00
04:00
03:00 Response data gets removed
04:00 28th request
Periodic request detected, client caches response and sends start polling request with interval set (e.g., to 1 hour) to server
2206
30/11/2010 05:00

FIG. 22

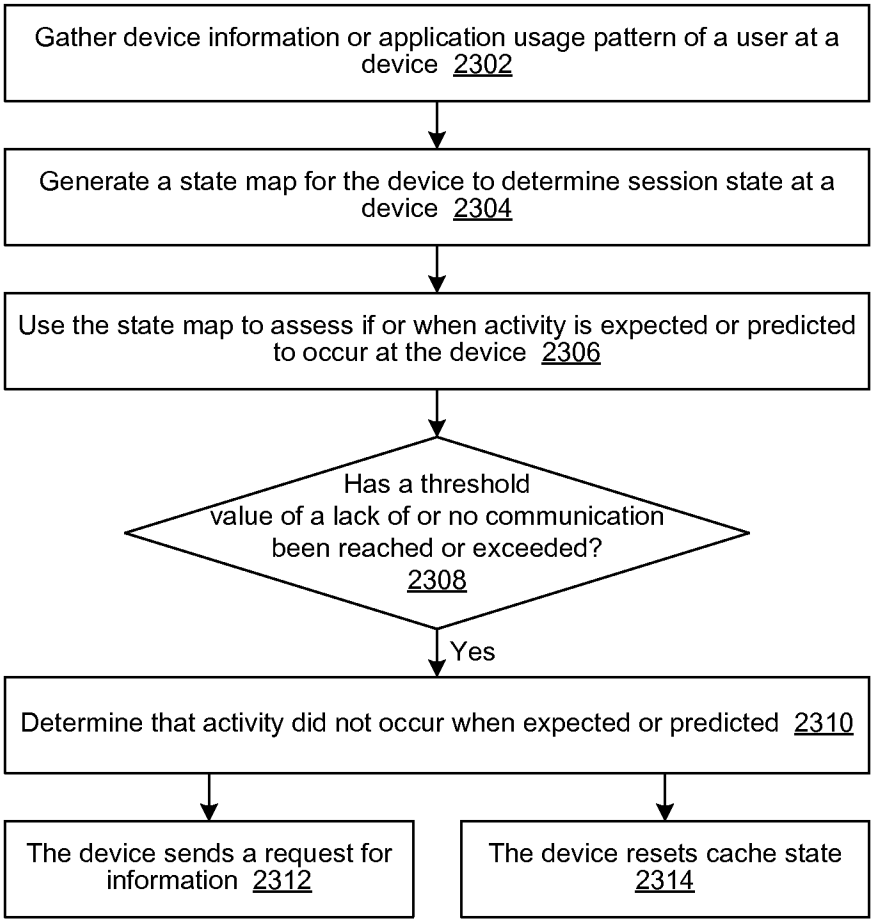

Gather device information or application usage pattern of a user at a device 2302
Generate a state map for the device to determine session state at a device 2304
Use the state map to assess if or when activity is expected or predicted to occur at the device 2306
Has a threshold value of a lack of or no communication been reached or exceeded? 2308
Yes
Determine that activity did not occur when expected or predicted 2310
The device sends a request for information 2312
The device resets cache state 2314

FIG. 23

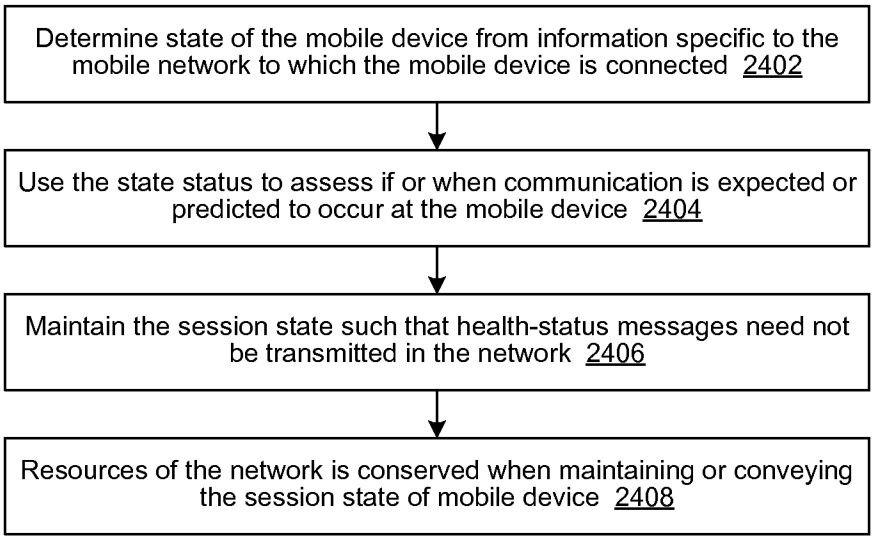

Determine state of the mobile device from information specific to the mobile network to which the mobile device is connected 2402
Use the state status to assess if or when communication is expected or predicted to occur at the mobile device 2404
Maintain the session state such that health-status messages need not be transmitted in the network 2406
Resources of the network is conserved when maintaining or conveying the session state of mobile device 2408

FIG. 24

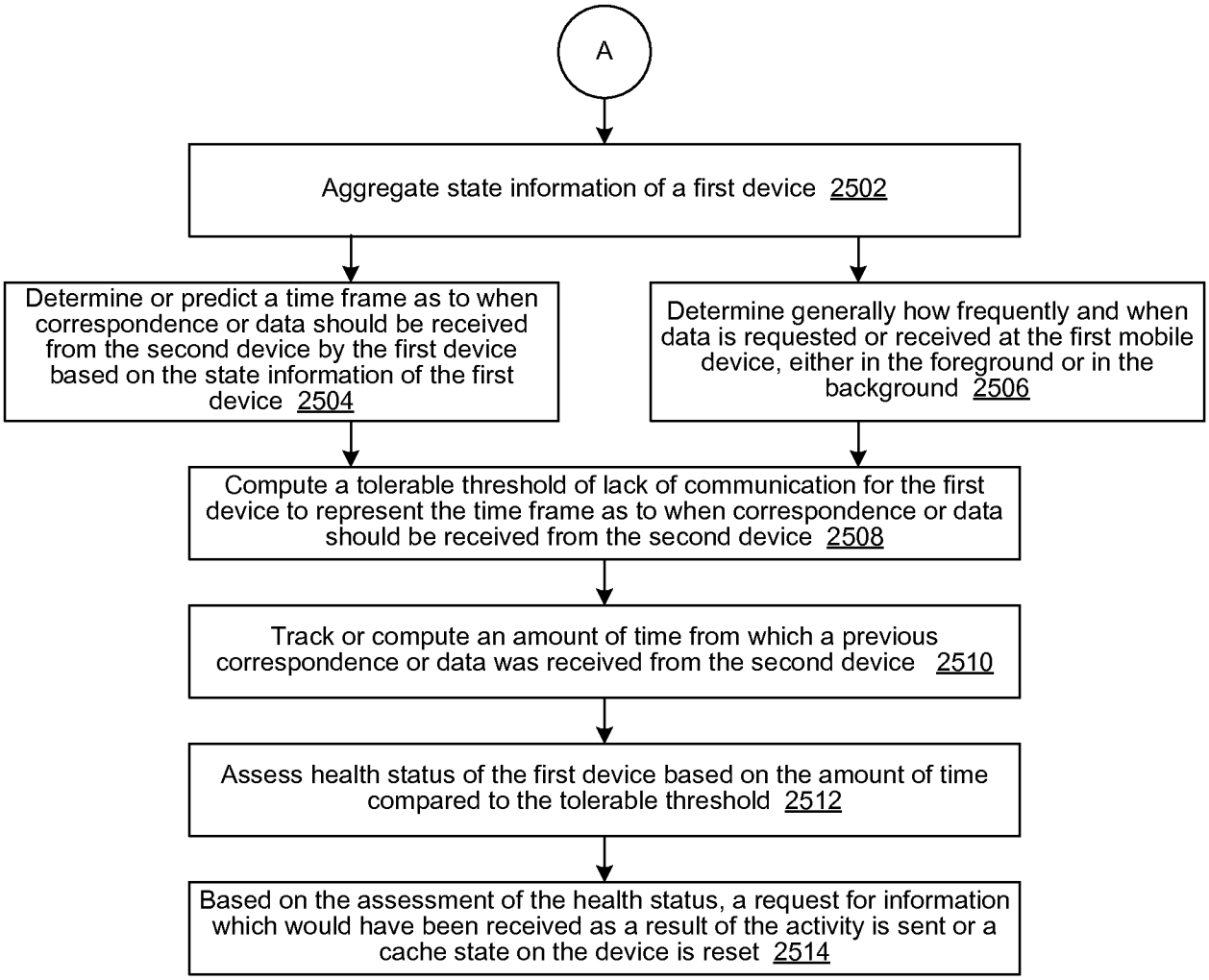

A
Aggregate state information of a first device 2502
Determine or predict a time frame as to when correspondence or data should be received from the second device by the first device based on the state information of the first device 2504
Determine generally how frequently and when data is requested or received at the first mobile device, either in the foreground or in the background 2506
Compute a tolerable threshold of lack of communication for the first device to represent the time frame as to when correspondence or data should be received from the second device 2508
Track or compute an amount of time from which a previous correspondence or data was received from the second device 2510
Assess health status of the first device based on the amount of time compared to the tolerable threshold 2512
Based on the assessment of the health status, a request for information which would have been received as a result of the activity is sent or a cache state on the device is reset 2514

FIG. 25

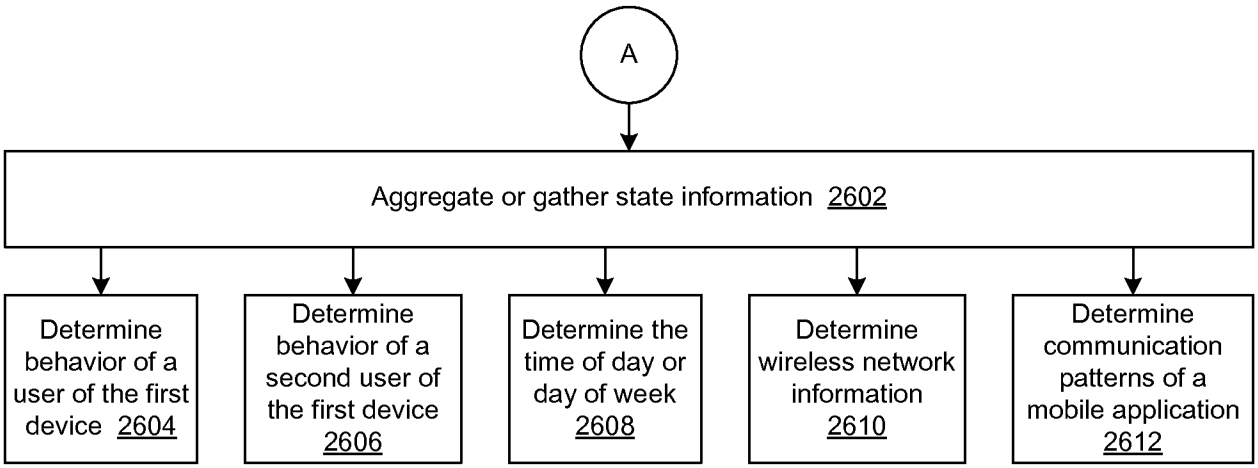

A
Aggregate or gather state information 2602
Determine behavior of a user of the first device 2604
Determine behavior of a second user of the first device 2606
Determine the time of day or day of week 2608
Determine wireless network information 2610
Determine communication patterns of a mobile application 2612

FIG. 25

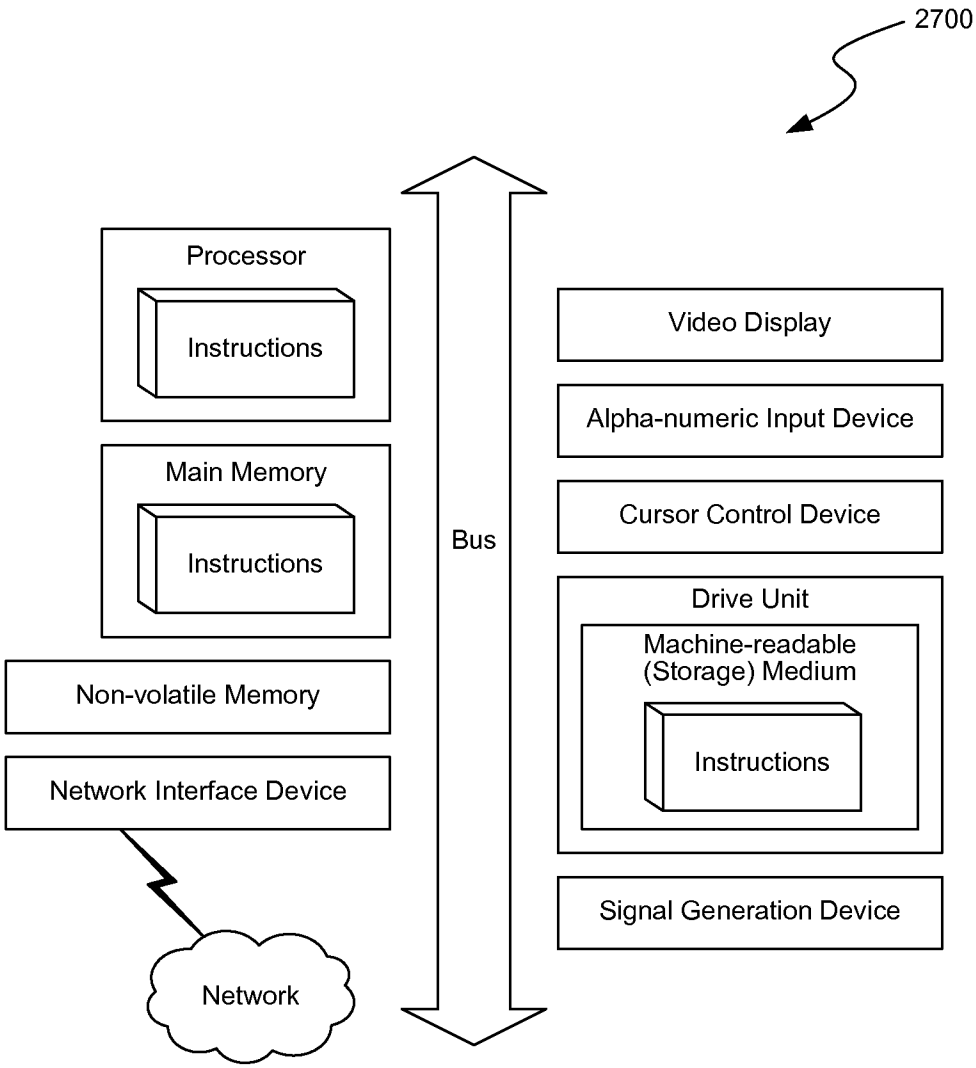

2700
Processor
Instructions
Main Memory
Instructions
Non-volatile Memory
Network Interface Device
Network
Bus
Video Display
Alpha-numeric Input Device
Cursor Control Device
Drive Unit
Machine-readable (Storage) Medium
Instructions
Signal Generation Device

FIG. 27

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/035257

A. CLASSIFICATION OF SUBJECT MATTER

H04W 24/04(2009.01)i, H04W 76/00(2009.01)i

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)
H04W 24/04; G06F 15/16; H04W 24/00; H04W 4/00; G06F 1/32; G06F 1/16; G06N 5/02; H04W 76/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: device, usage, state, map, predict, cache, conserve, message, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 2395412 A1 (RESEARCH IN MOTION LIMITED) 14 December 2011 See paragraphs [0068]-[0070]; claims 1, 13; and figure 4.	1,3,13,47-48
A		2,4-12,14-46,49-50
A	US 2010-0197282 A1 (KAORU UCHIDA) 5 August 2010 See paragraphs [0004], [0103]-[0108]; claim 1; and figure 6.	1-50
A	US 2009-0233595 A1 (JOHN M. HARRIS et al.) 17 September 2009 See paragraphs [0041]-[0048]; claim 1; and figure 6.	1-50
A	US 2010-0262664 A1 (MICHAEL BROWN et al.) 14 October 2010 See paragraphs [0047]-[0049]; claim 1; and figure 4.	1-50
A	US 2011-0119217 A1 (AE-KYEUNG MOON et al.) 19 May 2011 See paragraphs [0054]-[0055]; claim 1; and figure 5.	1-50

☐ 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 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	Date of mailing of the international search report
26 July 2013 (26.07.2013)	**26 July 2013 (26.07.2013)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
KIM Byoung Sung
Telephone No. +82-42-481-8403

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2013/035257

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 2395412 A1	14/12/2011	CA 2742534 A1	11/12/2011
US 2010-0197282 A1	05/08/2010	CN 101689133 A	31/03/2010
		CN 101689133 B	02/01/2013
		EP 2161665 A1	10/03/2010
		EP 2161665 A4	12/12/2012
		JP 5146453 B2	20/02/2013
		US 8265608 B2	11/09/2012
		WO 2009-001659 A1	31/12/2008
US 2009-0233595 A1	17/09/2009	None	
US 2010-0262664 A1	14/10/2010	CA 2690425 A1	08/10/2010
		CA 2690588 A1	08/10/2010
		EP 2239986 A1	13/10/2010
		EP 2239986 B1	21/11/2012
		EP 2239987 A1	13/10/2010
		US 2010-0262579 A1	14/10/2010
		US 8429236 B2	23/04/2013
US 2011-0119217 A1	19/05/2011	KR 10-1270747 B1	03/06/2013
		KR 10-2011-0055167 A	25/05/2011