US009325662B2

(12) **United States Patent**
Luna et al.

(10) **Patent No.: US 9,325,662 B2**
(45) **Date of Patent: Apr. 26, 2016**

(54) **SYSTEM AND METHOD FOR REDUCTION OF MOBILE NETWORK TRAFFIC USED FOR DOMAIN NAME SYSTEM (DNS) QUERIES**

(75) Inventors: **Michael Luna**, San Jose, CA (US); **Anton Diederich**, Mountain View, CA (US)

(73) Assignee: **Seven Networks, LLC**, Marshall, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 67 days.

(21) Appl. No.: **13/346,627**

(22) Filed: **Jan. 9, 2012**

(65) **Prior Publication Data**

US 2012/0179801 A1 Jul. 12, 2012

Related U.S. Application Data

(60) Provisional application No. 61/430,828, filed on Jan. 7, 2011, provisional application No. 61/533,007, filed on Sep. 9, 2011, provisional application No. 61/533,021, filed on Sep. 9, 2011.

(51) **Int. Cl.**
H04L 29/12 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 61/1511*** (2013.01); ***H04L 61/6009*** (2013.01)

(58) **Field of Classification Search**
CPC H04L 63/08; H04L 61/1511
USPC 709/223; 707/999.2
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

222,458	A	12/1879	Connolly et al.
447,918	A	3/1891	Strowger
4,200,770	A	4/1980	Hellman et al.
4,255,796	A	3/1981	Gabbe et al.
4,276,597	A	6/1981	Dissly et al.
4,531,020	A	7/1985	Wechselberger et al.
4,807,182	A	2/1989	Queen
4,831,582	A	5/1989	Miller et al.
4,875,159	A	10/1989	Cary et al.
4,897,781	A	1/1990	Chang et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2658185	A1	9/2009
EP	0772327	A2	5/1997

(Continued)

OTHER PUBLICATIONS

Pei et al., "Mobility Management in Hierarchical Multi-hop Mobile Wireless Networks", 1999.*

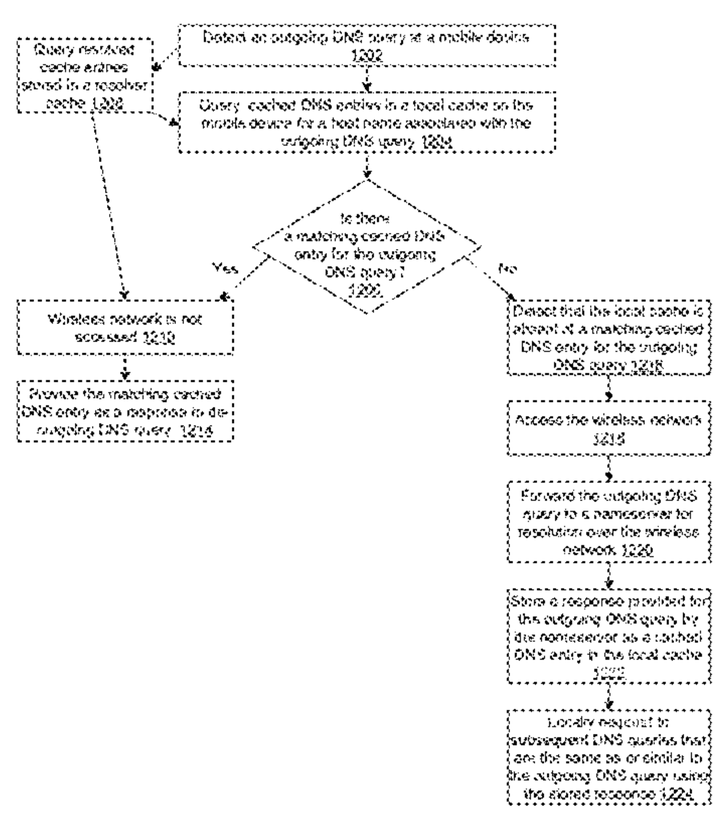
(Continued)

Primary Examiner — O. C. Vostal
(74) *Attorney, Agent, or Firm* — NK Patent Law, PLLC

(57) **ABSTRACT**

Systems and methods for reduction of mobile network traffic used for domain name system (DNS) queries are disclosed. In one embodiment, the method, which may be implemented on a system, includes, querying, cached DNS entries in a local cache on a mobile device for a host name associated with an outgoing DNS query from the mobile device or providing a matching cached DNS entry as a response to the outgoing DNS query without accessing the wireless network. The matching cached DNS entry can be invalidated in the local cache when a proxy remote from the mobile device detects a changed DNS response for the outgoing DNS query.

24 Claims, 26 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

4,972,457 A 11/1990 O'Sullivan
5,008,853 A 4/1991 Bly et al.
5,159,624 A 10/1992 Makita
5,220,657 A 6/1993 Bly et al.
5,263,157 A 11/1993 Janis
5,283,856 A 2/1994 Gross et al.
5,357,431 A 10/1994 Nakada et al.
5,384,892 A 1/1995 Strong
5,386,564 A 1/1995 Shearer et al.
5,392,390 A 2/1995 Crozier
5,434,994 A 7/1995 Shaheen et al.
5,436,960 A 7/1995 Campana, Jr. et al.
5,438,611 A 8/1995 Campana, Jr. et al.
5,479,472 A 12/1995 Campana, Jr. et al.
5,487,100 A 1/1996 Kane
5,491,703 A 2/1996 Barnaby et al.
5,493,692 A 2/1996 Theimer et al.
5,519,606 A 5/1996 Frid-Nielsen et al.
5,537,464 A 7/1996 Lewis et al.
5,555,376 A 9/1996 Theimer et al.
5,559,800 A 9/1996 Mousseau et al.
5,572,571 A 11/1996 Shirai
5,572,643 A 11/1996 Judson
5,574,859 A 11/1996 Yeh
5,581,749 A 12/1996 Hossain et al.
5,600,834 A 2/1997 Howard
5,603,054 A 2/1997 Theimer et al.
5,604,788 A 2/1997 Tett
5,613,012 A 3/1997 Hoffman et al.
5,619,507 A 4/1997 Tsuda
5,619,648 A 4/1997 Canale et al.
5,623,601 A 4/1997 Vu
5,625,670 A 4/1997 Campana, Jr. et al.
5,625,815 A 4/1997 Maier et al.
5,627,658 A 5/1997 Connors et al.
5,630,081 A 5/1997 Rybicki et al.
5,631,946 A 5/1997 Campana, Jr. et al.
5,632,018 A 5/1997 Otorii
5,634,053 A 5/1997 Noble et al.
5,644,788 A 7/1997 Courtright et al.
5,647,002 A 7/1997 Brunson
5,652,884 A 7/1997 Palevich
5,664,207 A 9/1997 Crumpler et al.
5,666,530 A 9/1997 Clark et al.
5,666,553 A 9/1997 Crozier
5,673,322 A 9/1997 Pepe et al.
5,680,542 A 10/1997 Mulchandani et al.
5,682,524 A 10/1997 Freund et al.
5,684,990 A 11/1997 Boothby
5,689,654 A 11/1997 Kikinis et al.
5,692,039 A 11/1997 Brankley et al.
5,696,903 A 12/1997 Mahany
5,701,423 A 12/1997 Crozier
5,701,469 A 12/1997 Brandli et al.
5,704,029 A 12/1997 Wright, Jr.
5,706,211 A 1/1998 Beletic et al.
5,706,502 A 1/1998 Foley et al.
5,706,507 A 1/1998 Schloss
5,710,918 A 1/1998 Lagarde et al.
5,713,019 A 1/1998 Keaten
5,715,403 A 2/1998 Stefik
5,717,925 A 2/1998 Harper et al.
5,721,908 A 2/1998 Lagarde et al.
5,721,914 A 2/1998 DeVries
5,727,202 A 3/1998 Kucala
5,729,549 A 3/1998 Kostreski et al.
5,729,704 A 3/1998 Stone et al.
5,729,735 A 3/1998 Meyering
5,742,905 A 4/1998 Pepe et al.
5,745,360 A 4/1998 Leone et al.
5,752,186 A 5/1998 Malackowski et al.
5,752,246 A 5/1998 Rogers et al.
5,754,938 A 5/1998 Herz et al.
5,757,916 A 5/1998 MacDoran et al.
5,758,088 A 5/1998 Bezaire et al.
5,758,150 A 5/1998 Bell et al.
5,758,322 A 5/1998 Rongley
5,758,354 A 5/1998 Huang et al.
5,758,355 A 5/1998 Buchanan
5,765,171 A 6/1998 Gehani et al.
5,778,346 A 7/1998 Frid-Nielsen et al.
5,778,361 A 7/1998 Nanjo et al.
5,781,614 A 7/1998 Brunson
5,781,901 A 7/1998 Kuzma
5,781,906 A 7/1998 Aggarwal et al.
5,787,430 A 7/1998 Doeringer et al.
5,787,441 A 7/1998 Beckhardt
5,790,425 A 8/1998 Wagle
5,790,790 A 8/1998 Smith et al.
5,790,974 A 8/1998 Tognazzini
5,793,413 A 8/1998 Hylton et al.
5,794,210 A 8/1998 Goldhaber et al.
5,799,318 A 8/1998 Cardinal et al.
5,802,312 A 9/1998 Lazaridis et al.
5,802,454 A 9/1998 Goshay et al.
5,802,518 A 9/1998 Karaev et al.
5,802,524 A 9/1998 Flowers et al.
5,806,074 A 9/1998 Souder et al.
5,809,242 A 9/1998 Shaw et al.
5,809,415 A 9/1998 Rossmann
5,818,437 A 10/1998 Grover et al.
5,819,172 A 10/1998 Campana, Jr. et al.
5,819,274 A 10/1998 Jackson, Jr.
5,819,284 A 10/1998 Farber et al.
5,822,324 A 10/1998 Kostresti et al.
5,822,747 A 10/1998 Graefe et al.
5,826,269 A 10/1998 Hussey
5,831,664 A 11/1998 Wharton et al.
5,832,483 A 11/1998 Barker
5,832,489 A 11/1998 Kucala
5,832,500 A 11/1998 Burrows
5,835,087 A 11/1998 Herz et al.
5,835,722 A 11/1998 Bradshaw et al.
5,838,252 A 11/1998 Kikinis
5,838,768 A 11/1998 Sumar et al.
5,838,973 A 11/1998 Carpenter-Smith et al.
5,845,278 A 12/1998 Kirsch et al.
5,852,775 A 12/1998 Hidary
5,852,820 A 12/1998 Burrows
5,857,201 A 1/1999 Wright, Jr. et al.
5,862,223 A 1/1999 Walker et al.
5,867,665 A 2/1999 Butman et al.
5,867,817 A 2/1999 Catallo et al.
5,870,759 A 2/1999 Bauer et al.
5,884,323 A 3/1999 Hawkins et al.
5,889,845 A 3/1999 Staples et al.
5,890,147 A 3/1999 Peltonen et al.
5,892,909 A 4/1999 Grasso et al.
5,898,780 A 4/1999 Liu et al.
5,898,917 A 4/1999 Batni et al.
5,903,723 A 5/1999 Beck et al.
5,907,618 A 5/1999 Gennaro et al.
5,909,689 A 6/1999 Van Ryzin
5,913,032 A 6/1999 Schwartz et al.
5,924,096 A 7/1999 Draper et al.
5,928,325 A 7/1999 Shaughnessy et al.
5,928,329 A 7/1999 Clark et al.
5,937,161 A 8/1999 Mulligan et al.
5,940,813 A 8/1999 Hutchings
5,943,676 A 8/1999 Boothby
5,948,066 A 9/1999 Whalen et al.
5,951,636 A 9/1999 Zerber
5,960,394 A 9/1999 Gould et al.
5,960,406 A 9/1999 Rasansky et al.
5,961,590 A 10/1999 Mendez et al.
5,963,642 A 10/1999 Goldstein
5,964,833 A 10/1999 Kikinis
5,968,131 A 10/1999 Mendez et al.
5,974,238 A 10/1999 Chase, Jr.
5,974,327 A 10/1999 Agrawal et al.
5,978,837 A 11/1999 Foladare et al.
5,978,933 A 11/1999 Wyld et al.
5,987,440 A 11/1999 O'Neil et al.
5,991,810 A * 11/1999 Shapiro et al. 709/229

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,000,000 A 12/1999 Hawkins et al.
6,003,070 A 12/1999 Frantz
6,006,197 A 12/1999 d'Eon et al.
6,006,274 A 12/1999 Hawkins et al.
6,016,478 A 1/2000 Zhang et al.
6,016,520 A 1/2000 Facq et al.
6,018,762 A 1/2000 Brunson et al.
6,023,700 A 2/2000 Owens et al.
6,023,708 A 2/2000 Mendez et al.
6,029,238 A 2/2000 Furukawa
6,034,621 A 3/2000 Kaufman
6,035,104 A 3/2000 Zahariev
6,044,372 A 3/2000 Rothfus et al.
6,044,381 A 3/2000 Boothby et al.
6,047,051 A 4/2000 Ginzboorg et al.
6,047,327 A 4/2000 Tso et al.
6,052,563 A 4/2000 Macko
6,052,735 A 4/2000 Ulrich et al.
6,057,855 A 5/2000 Barkans
6,065,055 A 5/2000 Hughes et al.
6,073,138 A 6/2000 de l'Etraz et al.
6,073,142 A 6/2000 Geiger et al.
6,073,165 A 6/2000 Narasimhan et al.
6,085,166 A 7/2000 Beckhardt et al.
6,085,192 A 7/2000 Mendez et al.
6,088,677 A 7/2000 Spurgeon
6,098,172 A 8/2000 Coss et al.
6,101,320 A 8/2000 Schuetze et al.
6,101,480 A 8/2000 Conmy et al.
6,101,531 A 8/2000 Eggleston et al.
6,112,181 A 8/2000 Shear et al.
6,119,014 A 9/2000 Alperovich et al.
6,119,171 A 9/2000 Alkhatib
6,125,369 A 9/2000 Wu et al.
6,125,388 A 9/2000 Reisman
6,128,627 A 10/2000 Mattis et al.
6,130,898 A 10/2000 Kostreski et al.
6,131,096 A 10/2000 Ng et al.
6,131,116 A 10/2000 Riggins et al.
6,134,432 A 10/2000 Holmes et al.
6,138,013 A 10/2000 Blanchard et al.
6,138,124 A 10/2000 Beckhardt
6,138,128 A 10/2000 Perkowitz et al.
6,138,146 A 10/2000 Moon et al.
6,141,664 A 10/2000 Boothby
6,151,606 A 11/2000 Mendez
6,157,630 A 12/2000 Adler et al.
6,161,008 A * 12/2000 Lee et al. 455/415
6,161,140 A 12/2000 Moriya
6,167,379 A 12/2000 Dean et al.
6,167,435 A 12/2000 Druckenmiller et al.
6,170,014 B1 1/2001 Darago et al.
6,173,312 B1 1/2001 Atarashi et al.
6,173,446 B1 1/2001 Khan et al.
6,175,831 B1 1/2001 Weinreich et al.
6,178,419 B1 1/2001 Legh-Smith et al.
6,181,935 B1 1/2001 Gossman et al.
6,185,184 B1 2/2001 Mattaway et al.
6,195,533 B1 2/2001 Tkatch et al.
6,198,696 B1 3/2001 Korpi et al.
6,198,922 B1 3/2001 Baynham
6,201,469 B1 3/2001 Balch et al.
6,202,085 B1 3/2001 Benson et al.
6,205,448 B1 3/2001 Kruglikov et al.
6,209,038 B1 3/2001 Bowen et al.
6,212,529 B1 4/2001 Boothby et al.
6,219,694 B1 4/2001 Lazaridis et al.
6,219,715 B1 * 4/2001 Ohno H04L 29/12216 370/401
6,221,877 B1 4/2001 Aronov et al.
6,223,187 B1 4/2001 Boothby et al.
6,226,686 B1 5/2001 Rothschild et al.
6,233,341 B1 5/2001 Riggins
6,243,705 B1 6/2001 Kucala
6,246,875 B1 6/2001 Seazholtz et al.
6,247,135 B1 6/2001 Feague
6,249,808 B1 6/2001 Seshadri
6,256,666 B1 7/2001 Singhal
6,263,201 B1 7/2001 Hashimoto et al.
6,263,340 B1 7/2001 Green
6,269,369 B1 7/2001 Robertson
6,272,545 B1 8/2001 Flanagin et al.
6,275,850 B1 8/2001 Beyda et al.
6,275,858 B1 8/2001 Bates et al.
6,289,212 B1 9/2001 Stein et al.
6,289,214 B1 9/2001 Backstrom
6,292,904 B1 9/2001 Broomhall et al.
6,295,541 B1 9/2001 Bodnar et al.
6,300,947 B1 10/2001 Kanevsky
6,304,881 B1 10/2001 Halim et al.
6,308,201 B1 10/2001 Pivowar et al.
6,311,216 B1 10/2001 Smith et al.
6,317,594 B1 11/2001 Gossman et al.
6,320,943 B1 11/2001 Borland
6,324,541 B1 11/2001 de l'Etraz et al.
6,324,542 B1 11/2001 Wright, Jr. et al.
6,324,544 B1 11/2001 Alam et al.
6,324,587 B1 11/2001 Trenbeath et al.
6,327,586 B1 12/2001 Kisiel
6,336,117 B1 1/2002 Massarani
6,336,138 B1 1/2002 Caswell et al.
6,341,311 B1 1/2002 Smith et al.
6,351,767 B1 2/2002 Batchelder et al.
6,356,937 B1 3/2002 Montville et al.
6,363,051 B1 3/2002 Eslambolchi et al.
6,363,352 B1 3/2002 Dailey et al.
6,370,566 B2 4/2002 Discolo et al.
6,377,790 B1 4/2002 Ishii
6,377,810 B1 4/2002 Geiger et al.
6,377,991 B1 4/2002 Smith et al.
6,380,959 B1 4/2002 Wang et al.
6,389,422 B1 5/2002 Doi et al.
6,389,455 B1 5/2002 Fuisz
6,389,457 B2 5/2002 Lazaridis et al.
6,397,057 B1 5/2002 Malackowski et al.
6,397,230 B1 5/2002 Carmel et al.
6,401,104 B1 6/2002 LaRue et al.
6,401,112 B1 6/2002 Boyer et al.
6,401,113 B2 6/2002 Lazaridis et al.
6,405,197 B2 6/2002 Gilmour
6,411,696 B1 6/2002 Iverson et al.
6,415,031 B1 7/2002 Colligan et al.
6,418,308 B1 7/2002 Heinonen et al.
6,421,669 B1 7/2002 Gilmour et al.
6,421,674 B1 7/2002 Yoakum et al.
6,421,781 B1 7/2002 Fox et al.
6,430,602 B1 8/2002 Kay et al.
6,438,585 B2 8/2002 Mousseau et al.
6,438,612 B1 8/2002 Ylonen et al.
6,442,589 B1 8/2002 Takahashi et al.
6,442,637 B1 8/2002 Hawkins et al.
6,446,118 B1 9/2002 Gottlieb
6,463,463 B1 10/2002 Godfrey et al.
6,463,464 B1 10/2002 Lazaridis et al.
6,484,203 B1 11/2002 Porras et al.
6,487,557 B1 11/2002 Nagatomo
6,487,560 B1 11/2002 LaRue et al.
6,490,353 B1 12/2002 Tan
6,496,802 B1 12/2002 van Zoest et al.
6,499,054 B1 12/2002 Hesselink et al.
6,505,214 B1 1/2003 Sherman et al.
6,516,327 B1 2/2003 Zondervan et al.
6,526,433 B1 2/2003 Chang et al.
6,526,506 B1 2/2003 Lewis
6,529,908 B1 3/2003 Piett et al.
6,532,446 B1 3/2003 King
6,535,892 B1 3/2003 LaRue et al.
6,546,005 B1 4/2003 Berkley et al.
6,549,939 B1 4/2003 Ford et al.
6,556,217 B1 4/2003 Mäkipää et al.
6,593,944 B1 7/2003 Nicolas et al.
6,601,026 B2 7/2003 Appelt et al.
6,615,253 B1 9/2003 Bowman-Amuah
6,618,710 B1 9/2003 Zondervan et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,621,892 B1 9/2003 Banister et al.
6,622,157 B1 9/2003 Heddaya et al.
6,625,621 B2 9/2003 Tan et al.
6,636,482 B2 10/2003 Cloonan et al.
6,639,693 B1 10/2003 Ejiri et al.
6,640,097 B2 10/2003 Corrigan et al.
6,640,244 B1 10/2003 Bowman-Amuah
6,640,249 B1 10/2003 Bowman-Amuah
6,643,650 B1 11/2003 Slaughter et al.
6,643,688 B1 11/2003 Fuisz
6,647,384 B2 11/2003 Gilmour
6,650,890 B1 11/2003 Irlam et al.
6,662,016 B1 12/2003 Buckham et al.
6,668,046 B1 12/2003 Albal
6,671,695 B2 12/2003 McFadden
6,671,700 B1 12/2003 Creemer et al.
6,671,702 B2 12/2003 Kruglikov et al.
6,671,757 B1 12/2003 Multer et al.
6,694,336 B1 2/2004 Multer et al.
6,697,807 B2 2/2004 McGeachie
6,701,378 B1 3/2004 Gilhuly et al.
6,707,801 B2 3/2004 Hsu
6,708,221 B1 3/2004 Mendez et al.
6,714,965 B2 3/2004 Kakuta et al.
6,721,787 B1 4/2004 Hiscock
6,727,917 B1 4/2004 Chew et al.
6,728,530 B1 4/2004 Heinonen et al.
6,728,786 B2 4/2004 Hawkins et al.
6,732,101 B1 5/2004 Cook
6,732,158 B1 5/2004 Hesselink et al.
6,735,591 B2 5/2004 Khan
6,741,232 B1 5/2004 Siedlikowski et al.
6,741,855 B1 5/2004 Martin et al.
6,742,015 B1 5/2004 Bowman-Amuah
6,742,059 B1 5/2004 Todd et al.
6,745,024 B1 6/2004 DeJaco et al.
6,745,326 B1 6/2004 Wary
6,756,882 B2 6/2004 Benes et al.
6,757,362 B1 6/2004 Cooper et al.
6,757,696 B2 6/2004 Multer et al.
6,757,708 B1 6/2004 Craig et al.
6,760,916 B2 7/2004 Holtz et al.
6,771,294 B1 8/2004 Pulli et al.
6,775,362 B1 8/2004 Ransom
6,779,019 B1 8/2004 Mousseau et al.
6,782,409 B1 8/2004 Yoshida
6,785,868 B1 8/2004 Raff
6,785,906 B1 8/2004 Gaughan et al.
6,799,190 B1 9/2004 Boothby
6,804,707 B1 10/2004 Ronning
6,810,405 B1 10/2004 LaRue et al.
6,816,849 B1 11/2004 Halt, Jr.
6,820,088 B1 11/2004 Hind et al.
6,820,204 B1 11/2004 Desai et al.
6,829,487 B2 12/2004 Eiden et al.
6,834,195 B2 12/2004 Brandenberg et al.
6,845,383 B1 1/2005 Kraenzel et al.
6,847,974 B2 1/2005 Wachtel
6,850,757 B2 2/2005 Watanabe et al.
6,857,009 B1 * 2/2005 Ferreria et al. 709/219
6,859,212 B2 2/2005 Kumar et al.
6,867,774 B1 3/2005 Halmshaw et al.
6,868,447 B1 3/2005 Slaughter et al.
6,871,220 B1 3/2005 Rajan et al.
6,871,236 B2 3/2005 Fishman et al.
6,873,688 B1 3/2005 Aarnio
6,874,017 B1 3/2005 Inoue et al.
6,879,985 B2 4/2005 Deguchi et al.
6,886,030 B1 4/2005 Easterbrook et al.
6,892,070 B2 5/2005 Warrier et al.
6,892,196 B1 5/2005 Hughes
6,895,394 B1 5/2005 Kremer et al.
6,895,558 B1 5/2005 Loveland
6,898,427 B1 5/2005 Griffith et al.
6,912,562 B1 6/2005 Krishnamurthy et al.
6,922,547 B2 7/2005 O'Neill et al.
6,922,721 B1 7/2005 Minborg et al.
6,925,477 B1 8/2005 Champagne et al.
6,931,529 B2 8/2005 Kunzinger
6,934,274 B2 * 8/2005 Inoue H04L 29/12216 370/338
6,938,079 B1 8/2005 Anderson et al.
6,941,351 B2 9/2005 Vetrivelkumaran et al.
6,944,447 B2 9/2005 Portman et al.
6,944,662 B2 9/2005 Devine et al.
6,947,770 B2 9/2005 Rydbeck
6,954,754 B2 10/2005 Peng
6,957,397 B1 10/2005 Hawkins et al.
6,965,917 B1 11/2005 Aloni et al.
6,965,968 B1 11/2005 Touboul
6,966,058 B2 11/2005 Earl et al.
6,968,175 B2 11/2005 Raivisto et al.
6,970,879 B1 11/2005 Gilmour
6,972,682 B2 12/2005 Lareau et al.
6,973,299 B2 12/2005 Apfel
6,981,041 B2 12/2005 Araujo et al.
6,981,047 B2 12/2005 Hanson et al.
6,985,933 B1 1/2006 Singhal et al.
6,985,983 B2 1/2006 Pellegrino et al.
6,986,061 B1 1/2006 Kunzinger
6,987,734 B2 1/2006 Hundemer
6,990,472 B2 1/2006 Rosenhaft et al.
6,993,326 B2 1/2006 Link, II et al.
6,993,327 B2 1/2006 Mathis
6,996,627 B1 2/2006 Carden
6,999,753 B2 2/2006 Beckmann et al.
7,016,710 B2 3/2006 Carmeli et al.
7,020,685 B1 3/2006 Chen et al.
7,024,491 B1 4/2006 Hanmann et al.
7,026,984 B1 4/2006 Thandu et al.
7,032,242 B1 4/2006 Grabelsky et al.
7,035,630 B2 4/2006 Knowles
7,046,993 B2 5/2006 Haaramo et al.
7,047,202 B2 5/2006 Jaipuria et al.
7,062,024 B2 6/2006 Kreckel et al.
7,069,308 B2 6/2006 Abrams
7,072,678 B2 7/2006 Allison
7,079,499 B1 7/2006 Akhtar et al.
7,080,371 B1 7/2006 Arnaiz et al.
7,082,316 B2 7/2006 Eiden et al.
7,085,260 B2 * 8/2006 Karaul et al. 370/352
7,085,365 B2 8/2006 Kauppinen
7,096,030 B2 8/2006 Huomo
7,100,821 B2 9/2006 Rasti
7,103,432 B2 9/2006 Drader et al.
7,120,692 B2 10/2006 Hesselink et al.
7,120,928 B2 10/2006 Sheth et al.
7,124,320 B1 10/2006 Wipfel
7,127,492 B1 10/2006 Calo et al.
7,130,839 B2 10/2006 Boreham et al.
7,136,645 B2 11/2006 Hanson et al.
7,139,555 B2 11/2006 Apfel
7,139,565 B2 11/2006 Fiatal et al.
7,140,549 B2 11/2006 de Jong
7,146,645 B1 12/2006 Hellsten et al.
7,149,780 B2 12/2006 Quine et al.
7,149,789 B2 12/2006 Slivka et al.
7,149,959 B1 12/2006 Jones et al.
7,162,241 B2 1/2007 Kim et al.
7,165,727 B2 1/2007 de Jong
7,172,118 B2 2/2007 Urken
7,181,228 B2 2/2007 Boesch
7,184,790 B2 2/2007 Dorenbosch et al.
7,185,362 B2 2/2007 Hawkes et al.
7,194,273 B2 3/2007 Vaudreuil
7,200,390 B1 4/2007 Henager et al.
7,203,733 B1 4/2007 Bern
7,206,806 B2 4/2007 Pineau
7,209,757 B2 4/2007 Naghian et al.
7,210,121 B2 4/2007 Xia et al.
7,219,139 B2 5/2007 Martin et al.
7,219,222 B1 5/2007 Durbin et al.
7,224,957 B2 5/2007 Spector
7,231,206 B2 6/2007 Cudak et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,233,795 B1 6/2007 Ryden
7,234,111 B2 6/2007 Chu et al.
7,239,877 B2 7/2007 Corneille et al.
7,240,095 B1 7/2007 Lewis
7,242,680 B2 7/2007 Gallant
7,245,926 B2 7/2007 Liao et al.
7,257,391 B2 8/2007 Burgess et al.
7,257,639 B1 8/2007 Li et al.
7,259,666 B1 8/2007 Hermsmeyer et al.
7,260,552 B2 8/2007 Riera Jorba et al.
7,260,590 B1 8/2007 Williams
7,260,651 B2 8/2007 Parrella, Sr. et al.
7,272,830 B2 9/2007 de Jong
7,274,783 B2 9/2007 Yoakum et al.
7,277,408 B2 10/2007 Sorsa
7,284,664 B1 10/2007 Ivchenko et al.
7,289,519 B1 * 10/2007 Liskov H04L 29/12066 370/230
7,289,792 B1 10/2007 Turunen
7,289,964 B1 10/2007 Bowman-Amuah
7,289,971 B1 10/2007 O'Neil et al.
7,293,107 B1 11/2007 Hanson et al.
7,295,853 B2 11/2007 Jin et al.
7,296,155 B1 11/2007 Trostle et al.
7,298,708 B2 * 11/2007 Nasiri H04L 29/12009 370/254
7,305,252 B2 12/2007 Britt et al.
7,305,700 B2 12/2007 Boynton et al.
7,310,350 B1 12/2007 Shao et al.
7,310,729 B2 12/2007 Gordon et al.
7,324,473 B2 1/2008 Corneille et al.
7,333,459 B2 * 2/2008 Feyerabend H04L 29/06 370/338
7,337,219 B1 2/2008 Meenan et al.
7,343,396 B2 3/2008 Kausik et al.
7,349,871 B2 3/2008 Labrou et al.
7,353,274 B1 4/2008 Rouhi et al.
7,359,720 B2 4/2008 Hartmaier et al.
7,373,386 B2 5/2008 Gardner et al.
7,373,661 B2 5/2008 Smith et al.
7,374,099 B2 5/2008 de Jong
7,376,701 B2 5/2008 Bhargava et al.
7,382,879 B1 6/2008 Miller
7,383,339 B1 6/2008 Meenan et al.
7,388,950 B2 6/2008 Elsey et al.
7,389,412 B2 6/2008 Sharma et al.
7,392,483 B2 6/2008 Wong et al.
7,395,329 B1 7/2008 Holt et al.
7,398,271 B1 7/2008 Borkovsky et al.
7,430,609 B2 9/2008 Brown et al.
7,441,271 B2 10/2008 Fiatal et al.
7,443,847 B1 10/2008 Albert et al.
7,461,071 B2 12/2008 Fitzpatrick et al.
7,465,231 B2 12/2008 Lewin et al.
7,469,125 B2 12/2008 Nurmi
7,472,424 B2 12/2008 Evans et al.
7,483,036 B2 1/2009 Moore
7,499,537 B2 3/2009 Elsey et al.
7,499,998 B2 * 3/2009 Toebes et al. 709/225
7,502,615 B2 3/2009 Wilhoite et al.
7,519,042 B2 4/2009 Gorday et al.
7,532,571 B1 5/2009 Price et al.
7,539,665 B2 5/2009 Mendez
7,548,947 B2 6/2009 Kasriel et al.
7,548,969 B2 6/2009 Tripp et al.
7,551,900 B2 6/2009 Kang et al.
7,567,575 B2 7/2009 Chen et al.
7,574,208 B2 8/2009 Hanson et al.
7,574,499 B1 * 8/2009 Swildens et al. 709/223
7,575,171 B2 8/2009 Lev
7,584,294 B2 9/2009 Plamondon
7,587,482 B2 9/2009 Henderson et al.
7,587,608 B2 9/2009 Haller et al.
7,593,714 B2 9/2009 Schultz et al.
7,596,608 B2 9/2009 Alexander et al.
7,596,791 B2 9/2009 Wei et al.
7,613,792 B2 11/2009 Zervas et al.
7,630,986 B1 12/2009 Herz et al.
7,634,558 B1 12/2009 Mangal et al.
7,636,763 B1 12/2009 Fein et al.
7,643,818 B2 1/2010 Backholm et al.
7,644,166 B2 1/2010 Appelman et al.
7,647,047 B2 1/2010 Moghaddam et al.
7,650,416 B2 1/2010 Wu et al.
7,650,432 B2 1/2010 Bosworth et al.
7,672,291 B2 3/2010 Wang
7,672,439 B2 3/2010 Appelman et al.
7,680,281 B2 3/2010 Fiatal et al.
7,680,520 B2 3/2010 Ruuska et al.
7,684,346 B2 3/2010 Valli
7,689,664 B2 3/2010 Karlberg
7,693,555 B2 4/2010 Srinivasan et al.
7,693,944 B2 4/2010 Appelman et al.
7,694,008 B2 4/2010 Chang et al.
7,706,781 B2 4/2010 Backholm et al.
7,707,573 B1 4/2010 Marmaros et al.
7,720,936 B2 * 5/2010 Plamondon 709/219
7,725,602 B2 * 5/2010 Liu et al. 709/245
7,751,803 B2 7/2010 Vialen et al.
7,752,633 B1 7/2010 Fleming
7,757,956 B2 7/2010 Koenck et al.
7,769,395 B2 8/2010 Fiatal et al.
7,769,400 B2 8/2010 Backholm et al.
7,769,805 B1 8/2010 Barnes et al.
7,770,223 B2 8/2010 Shevenell et al.
7,778,792 B2 8/2010 Huang et al.
7,783,757 B2 8/2010 Plamondon
7,796,742 B1 9/2010 Sutaria et al.
7,797,064 B2 9/2010 Loomis et al.
7,809,818 B2 10/2010 Plamondon
7,827,055 B1 11/2010 Snodgrass et al.
7,827,597 B2 11/2010 Boynton et al.
7,849,507 B1 12/2010 Bloch et al.
7,853,563 B2 12/2010 Alvarado et al.
7,873,609 B2 1/2011 Kim et al.
7,877,703 B1 1/2011 Fleming
7,877,807 B2 1/2011 Shipp
7,881,745 B1 2/2011 Rao et al.
7,899,996 B1 3/2011 Levin-Michael
7,908,656 B1 3/2011 Mu
7,917,468 B2 3/2011 Ariel et al.
7,917,505 B2 3/2011 van Gent et al.
7,921,167 B2 4/2011 Shroff et al.
7,930,416 B2 4/2011 Miller et al.
7,933,929 B1 4/2011 McClendon et al.
7,937,091 B2 5/2011 Roman et al.
7,941,582 B2 5/2011 Bushell et al.
7,970,860 B2 6/2011 Kline et al.
7,970,878 B1 * 6/2011 Burshan H04L 63/0236 370/255
7,970,939 B1 * 6/2011 Satish et al. 709/245
7,996,487 B2 8/2011 Snyder
8,005,891 B2 8/2011 Knowles et al.
8,010,082 B2 8/2011 Sutaria et al.
8,032,409 B1 10/2011 Mikurak
8,060,154 B1 11/2011 Bali et al.
8,064,583 B1 11/2011 Sutaria et al.
8,069,166 B2 11/2011 Alvarado et al.
8,074,162 B1 12/2011 Cohen
8,078,158 B2 12/2011 Backholm
8,087,085 B2 12/2011 Hu et al.
8,107,921 B2 1/2012 Fiatal
8,116,214 B2 2/2012 Backholm et al.
8,127,342 B2 2/2012 Boynton et al.
8,131,763 B2 3/2012 Tuscano et al.
8,166,164 B1 4/2012 Luna et al.
8,190,701 B2 5/2012 Luna et al.
8,194,680 B1 6/2012 Brandwine et al.
8,195,196 B2 * 6/2012 Haran et al. 455/456.3
8,204,953 B2 6/2012 Luna et al.
8,209,709 B2 6/2012 Fleming
8,214,813 B2 7/2012 Harris et al.
8,239,549 B2 * 8/2012 Aura H04L 61/2015 709/228

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,239,915 B1 8/2012 Satish et al.
8,260,852 B1 9/2012 Cselle
8,291,076 B2 10/2012 Luna et al.
8,316,098 B2 11/2012 Luna et al.
8,326,985 B2 12/2012 Luna et al.
8,340,633 B1 12/2012 Rege et al.
8,356,080 B2 1/2013 Luna et al.
8,364,181 B2 1/2013 Backholm et al.
8,412,675 B2 4/2013 Alvarado et al.
8,417,823 B2 4/2013 Luna et al.
8,438,633 B1 5/2013 Backholm et al.
8,447,836 B2 * 5/2013 Mahalal et al. 709/219
8,484,314 B2 7/2013 Luna et al.
8,522,086 B1 * 8/2013 Garrett 714/48
2001/0005364 A1 6/2001 Kang
2001/0009025 A1 7/2001 Ahonen
2001/0010046 A1 7/2001 Muyres et al.
2001/0013069 A1 8/2001 Shah
2001/0023414 A1 9/2001 Kumar et al.
2001/0032254 A1 10/2001 Hawkins
2001/0034225 A1 10/2001 Gupte et al.
2001/0034244 A1 10/2001 Calder et al.
2001/0037453 A1 11/2001 Mitty et al.
2001/0039191 A1 11/2001 Maierhofer
2001/0041566 A1 11/2001 Xanthos et al.
2001/0042009 A1 11/2001 Montague
2001/0042099 A1 11/2001 Peng
2001/0043148 A1 11/2001 Stewart
2001/0052052 A1 12/2001 Peng
2001/0053687 A1 12/2001 Sivula
2002/0002478 A1 1/2002 Swart et al.
2002/0002591 A1 1/2002 Ketola
2002/0004746 A1 1/2002 Ferber et al.
2002/0007303 A1 1/2002 Brookler et al.
2002/0013727 A1 1/2002 Lee
2002/0019225 A1 2/2002 Miyashita
2002/0019812 A1 2/2002 Board et al.
2002/0019830 A1 2/2002 Hamberg et al.
2002/0021665 A1 * 2/2002 Bhagavath et al. 370/229
2002/0032671 A1 3/2002 Iinuma
2002/0035556 A1 3/2002 Shah et al.
2002/0035617 A1 3/2002 Lynch et al.
2002/0038253 A1 3/2002 Seaman et al.
2002/0042875 A1 4/2002 Shukla
2002/0049828 A1 4/2002 Pekarek-Kostka
2002/0049857 A1 4/2002 Farber et al.
2002/0052965 A1 5/2002 Dowling
2002/0053078 A1 5/2002 Holtz et al.
2002/0055351 A1 5/2002 Elsey et al.
2002/0059201 A1 5/2002 Work
2002/0059251 A1 5/2002 Stern et al.
2002/0059457 A1 5/2002 Ballard et al.
2002/0062467 A1 5/2002 Hunzinger
2002/0065110 A1 5/2002 Enns et al.
2002/0068559 A1 6/2002 Sharma et al.
2002/0069037 A1 6/2002 Hendrickson et al.
2002/0073207 A1 6/2002 Widger et al.
2002/0077077 A1 6/2002 Rezvani et al.
2002/0077084 A1 6/2002 Zellner et al.
2002/0078206 A1 * 6/2002 Boies et al. 709/226
2002/0078384 A1 6/2002 Hippelainen
2002/0087549 A1 7/2002 Mostafa
2002/0087661 A1 * 7/2002 Matichuk et al. 709/218
2002/0087679 A1 7/2002 Pulley et al.
2002/0087883 A1 7/2002 Wohlgemuth et al.
2002/0089542 A1 7/2002 Imamura
2002/0091921 A1 7/2002 Kunzinger
2002/0095319 A1 7/2002 Swart et al.
2002/0095328 A1 7/2002 Swart et al.
2002/0095391 A1 7/2002 Swart et al.
2002/0095399 A1 7/2002 Devine et al.
2002/0098855 A1 7/2002 Hartmaier et al.
2002/0099613 A1 7/2002 Swart et al.
2002/0099809 A1 7/2002 Lee
2002/0101975 A1 8/2002 Tiburtius et al.
2002/0103008 A1 8/2002 Rahn et al.
2002/0103934 A1 8/2002 Fishman et al.
2002/0107042 A1 8/2002 Murnaghan et al.
2002/0107944 A1 8/2002 Bai et al.
2002/0107985 A1 8/2002 Hwang et al.
2002/0116499 A1 8/2002 Ennus et al.
2002/0116501 A1 8/2002 Ho et al.
2002/0120388 A1 8/2002 Bullock
2002/0120766 A1 8/2002 Okajima et al.
2002/0120779 A1 8/2002 Teeple et al.
2002/0126701 A1 9/2002 Requena
2002/0128908 A1 9/2002 Levin et al.
2002/0133504 A1 9/2002 Vlahos et al.
2002/0144109 A1 10/2002 Benantar et al.
2002/0146129 A1 10/2002 Kaplan
2002/0152379 A1 10/2002 Gefwert et al.
2002/0155848 A1 10/2002 Suryanarayana
2002/0156839 A1 10/2002 Peterson et al.
2002/0158908 A1 10/2002 Vaajala et al.
2002/0161587 A1 10/2002 Pitts, III et al.
2002/0161925 A1 10/2002 Munger et al.
2002/0161928 A1 10/2002 Ndili
2002/0164977 A1 11/2002 Link, II et al.
2002/0165988 A1 11/2002 Khan et al.
2002/0167484 A1 11/2002 Hatanaka et al.
2002/0174189 A1 11/2002 Peng
2002/0186848 A1 12/2002 Shaik
2002/0188940 A1 12/2002 Breckner et al.
2002/0193094 A1 12/2002 Lawless et al.
2002/0194183 A1 12/2002 Yoakum et al.
2002/0194207 A1 12/2002 Bartlett et al.
2002/0194209 A1 12/2002 Bolosky et al.
2002/0198027 A1 12/2002 Rydbeck
2002/0198991 A1 12/2002 Gopalakrishnan et al.
2003/0005151 A1 1/2003 Ullman et al.
2003/0009571 A1 1/2003 Bavadekar
2003/0014491 A1 1/2003 Horvitz et al.
2003/0021400 A1 1/2003 Grandgent et al.
2003/0022662 A1 1/2003 Mittal
2003/0023692 A1 1/2003 Moroo
2003/0023975 A1 1/2003 Schrader et al.
2003/0028430 A1 2/2003 Zimmerman
2003/0028441 A1 2/2003 Barsness et al.
2003/0037094 A1 2/2003 Douceur et al.
2003/0046433 A1 3/2003 Luzzatti et al.
2003/0046586 A1 3/2003 Bheemarasetti et al.
2003/0046587 A1 3/2003 Bheemarasetti et al.
2003/0050041 A1 3/2003 Wu
2003/0051142 A1 3/2003 Hidalgo et al.
2003/0054810 A1 3/2003 Chen et al.
2003/0055555 A1 3/2003 Knockeart et al.
2003/0056096 A1 3/2003 Albert et al.
2003/0060188 A1 3/2003 Gidron et al.
2003/0063120 A1 4/2003 Wong et al.
2003/0065738 A1 4/2003 Yang et al.
2003/0065739 A1 4/2003 Shnier
2003/0065802 A1 4/2003 Vitikainen et al.
2003/0070061 A1 4/2003 Wong et al.
2003/0072451 A1 4/2003 Pimentel et al.
2003/0078880 A1 4/2003 Alley et al.
2003/0084165 A1 5/2003 Kjellberg et al.
2003/0084361 A1 5/2003 Lawrence et al.
2003/0088629 A1 5/2003 Berkowitz et al.
2003/0093691 A1 5/2003 Simon et al.
2003/0096608 A1 5/2003 Mortensen et al.
2003/0097381 A1 5/2003 Detweiler et al.
2003/0100321 A1 5/2003 Rao et al.
2003/0100326 A1 5/2003 Grube et al.
2003/0105837 A1 6/2003 Kamen et al.
2003/0117432 A1 6/2003 Kautto-Kiovula et al.
2003/0120685 A1 6/2003 Duncombe et al.
2003/0125023 A1 7/2003 Fishler
2003/0126216 A1 7/2003 Avila et al.
2003/0126233 A1 7/2003 Bryers et al.
2003/0130984 A1 7/2003 Quinlan et al.
2003/0145038 A1 7/2003 Bin Tariq et al.
2003/0146934 A1 8/2003 Bailey et al.
2003/0153338 A1 8/2003 Herz et al.
2003/0154212 A1 8/2003 Schirmer et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2003/0156146 A1 8/2003 Suomela et al.
2003/0157947 A1 8/2003 Fiatal et al.
2003/0169262 A1 9/2003 Lavelle et al.
2003/0172112 A1 9/2003 Vignaud
2003/0177281 A1 9/2003 McQuillan et al.
2003/0182420 A1 9/2003 Jones et al.
2003/0182424 A1* 9/2003 Odendahl 709/225
2003/0182431 A1 9/2003 Sturniolo et al.
2003/0187984 A1 10/2003 Banavar et al.
2003/0204605 A1 10/2003 Hudson et al.
2003/0204708 A1 10/2003 Hulme et al.
2003/0208529 A1 11/2003 Pendyala et al.
2003/0208559 A1 11/2003 Velline et al.
2003/0210666 A1 11/2003 Trossen et al.
2003/0211845 A1 11/2003 Lohtia et al.
2003/0217098 A1 11/2003 Bobde et al.
2003/0217142 A1 11/2003 Bobde et al.
2003/0223554 A1 12/2003 Zhang
2003/0227487 A1 12/2003 Hugh
2003/0227745 A1 12/2003 Khoo
2003/0233329 A1 12/2003 Laraki et al.
2003/0235308 A1 12/2003 Boynton et al.
2003/0236857 A1 12/2003 Takase et al.
2003/0236981 A1 12/2003 Marmigere et al.
2004/0002324 A1 1/2004 Juntunen et al.
2004/0006630 A1 1/2004 Friend et al.
2004/0010590 A1 1/2004 Manzano
2004/0015504 A1 1/2004 Ahad et al.
2004/0024795 A1 2/2004 Hind et al.
2004/0024892 A1 2/2004 Creswell et al.
2004/0027326 A1 2/2004 Hays et al.
2004/0027375 A1 2/2004 Ellis et al.
2004/0027378 A1 2/2004 Hays et al.
2004/0043770 A1 3/2004 Amit et al.
2004/0047356 A1 3/2004 Bauer
2004/0049579 A1 3/2004 Ims et al.
2004/0049599 A1 3/2004 Friend et al.
2004/0051715 A1 3/2004 Brokenshire et al.
2004/0054711 A1 3/2004 Multer
2004/0054739 A1 3/2004 Friend et al.
2004/0064445 A1 4/2004 Pfleging et al.
2004/0064488 A1 4/2004 Sinha
2004/0068579 A1 4/2004 Marmigere et al.
2004/0068698 A1 4/2004 Wu et al.
2004/0072559 A1 4/2004 Kakumaru et al.
2004/0073476 A1 4/2004 Donahue et al.
2004/0073651 A1 4/2004 Beaulieu et al.
2004/0073707 A1* 4/2004 Dillon 709/245
2004/0073867 A1 4/2004 Kausik et al.
2004/0075675 A1 4/2004 Raivisto et al.
2004/0075695 A1 4/2004 Chew et al.
2004/0078814 A1 4/2004 Allen
2004/0080515 A1 4/2004 Hagiwara
2004/0082346 A1 4/2004 Skytt et al.
2004/0085909 A1 5/2004 Soliman
2004/0098625 A1 5/2004 Lagadec et al.
2004/0103147 A1 5/2004 Flesher et al.
2004/0107319 A1 6/2004 D'Orto et al.
2004/0110497 A1 6/2004 Little
2004/0120323 A1 6/2004 Viikari et al.
2004/0122907 A1 6/2004 Chou et al.
2004/0123095 A1 6/2004 Marshall
2004/0123304 A1 6/2004 Black et al.
2004/0127214 A1 7/2004 Reddy et al.
2004/0128375 A1 7/2004 Rockwell
2004/0133626 A1 7/2004 Herrero et al.
2004/0141011 A1 7/2004 Smethers et al.
2004/0147248 A1 7/2004 Will
2004/0147262 A1 7/2004 Lescuyer et al.
2004/0148375 A1 7/2004 Levett et al.
2004/0151186 A1 8/2004 Akama
2004/0158611 A1 8/2004 Daniell et al.
2004/0162076 A1* 8/2004 Chowdry et al. 455/445
2004/0162890 A1 8/2004 Ohta
2004/0167966 A1 8/2004 Lee et al.
2004/0170257 A1 9/2004 Gross et al.
2004/0172481 A1 9/2004 Engstrom
2004/0176128 A1 9/2004 Grabelsky et al.
2004/0177369 A1 9/2004 Akins, III
2004/0179513 A1 9/2004 Smith et al.
2004/0181550 A1 9/2004 Warsta et al.
2004/0184475 A1 9/2004 Meier
2004/0186902 A1 9/2004 Stewart
2004/0189610 A1 9/2004 Friend
2004/0199497 A1 10/2004 Timmons
2004/0199582 A1 10/2004 Kucharewski et al.
2004/0199649 A1* 10/2004 Tarnanen et al. 709/230
2004/0199663 A1 10/2004 Horvitz et al.
2004/0204085 A1 10/2004 Vargas et al.
2004/0205248 A1 10/2004 Little et al.
2004/0205330 A1 10/2004 Godfrey et al.
2004/0209602 A1 10/2004 Joyce et al.
2004/0210639 A1 10/2004 Ben-Yoseph et al.
2004/0215665 A1* 10/2004 Edgar et al. 707/200
2004/0215823 A1* 10/2004 Kleinfelter et al. 709/245
2004/0218609 A1 11/2004 Foster et al.
2004/0219940 A1 11/2004 Kong et al.
2004/0229609 A1 11/2004 Yamaguchi
2004/0230619 A1 11/2004 Blanco et al.
2004/0233930 A1 11/2004 Colby, Jr.
2004/0236792 A1 11/2004 Celik
2004/0242209 A1 12/2004 Kruis et al.
2004/0252816 A1 12/2004 Nicolas
2004/0255126 A1 12/2004 Reith
2004/0258231 A1 12/2004 Elsey et al.
2004/0259535 A1 12/2004 Elsey et al.
2004/0259537 A1 12/2004 Ackley
2004/0260948 A1 12/2004 Miyata et al.
2004/0264396 A1 12/2004 Ginzburg et al.
2004/0266364 A1 12/2004 Nguyen et al.
2004/0268148 A1 12/2004 Karjala et al.
2005/0002501 A1 1/2005 Elsey et al.
2005/0002508 A1 1/2005 Elsey et al.
2005/0002509 A1 1/2005 Elsey et al.
2005/0002510 A1 1/2005 Elsey et al.
2005/0010694 A1 1/2005 Ma et al.
2005/0015432 A1 1/2005 Cohen
2005/0021750 A1 1/2005 Abrams
2005/0022000 A1 1/2005 Inomata et al.
2005/0022182 A1 1/2005 Mittal
2005/0027591 A9 2/2005 Gailey et al.
2005/0027716 A1 2/2005 Apfel
2005/0027869 A1 2/2005 Johnson
2005/0033657 A1 2/2005 Herrington et al.
2005/0033812 A1 2/2005 McCarthy et al.
2005/0033926 A1 2/2005 Dumont
2005/0037741 A1 2/2005 Gilbert
2005/0038707 A1 2/2005 Roever et al.
2005/0038724 A1 2/2005 Roever et al.
2005/0038863 A1 2/2005 Onyon et al.
2005/0041584 A1 2/2005 Lau et al.
2005/0041793 A1 2/2005 Fulton et al.
2005/0044144 A1 2/2005 Malik et al.
2005/0050222 A1 3/2005 Packer
2005/0054381 A1 3/2005 Lee et al.
2005/0055578 A1 3/2005 Wright et al.
2005/0063544 A1 3/2005 Uusitalo et al.
2005/0071489 A1 3/2005 Parupudi et al.
2005/0071511 A1 3/2005 Chen
2005/0071674 A1 3/2005 Chou et al.
2005/0073982 A1 4/2005 Corneille et al.
2005/0075109 A1 4/2005 Neyret et al.
2005/0076136 A1 4/2005 Cho et al.
2005/0076241 A1 4/2005 Appelman
2005/0080928 A1 4/2005 Beverly et al.
2005/0086385 A1 4/2005 Rouleau
2005/0086540 A1 4/2005 Gunter et al.
2005/0094625 A1 5/2005 Bouat
2005/0097225 A1 5/2005 Glatt et al.
2005/0097570 A1 5/2005 Bomers
2005/0099963 A1 5/2005 Multer et al.
2005/0101307 A1 5/2005 Brugge et al.
2005/0102257 A1 5/2005 Onyon et al.
2005/0102328 A1 5/2005 Ring et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2005/0102351 A1 5/2005 Jiang et al.
2005/0108322 A1 5/2005 Kline et al.
2005/0108427 A1 5/2005 Datta
2005/0117606 A1 6/2005 Kim
2005/0120082 A1 6/2005 Hesselink et al.
2005/0120084 A1 6/2005 Hu et al.
2005/0120181 A1 6/2005 Arunagirinathan et al.
2005/0122333 A1 6/2005 Sumanaweera et al.
2005/0124332 A1 6/2005 Clark et al.
2005/0125459 A1 6/2005 Sutinen et al.
2005/0138111 A1 6/2005 Aton et al.
2005/0138176 A1 6/2005 Singh et al.
2005/0138198 A1 6/2005 May
2005/0144219 A1 6/2005 Terada
2005/0147130 A1 7/2005 Hurwitz et al.
2005/0154698 A1 7/2005 Ikezawa et al.
2005/0154796 A1 7/2005 Forsyth
2005/0154836 A1 7/2005 Steeley et al.
2005/0155027 A1 7/2005 Wei
2005/0164703 A1 7/2005 Huynh
2005/0164721 A1 7/2005 Eric Yeh et al.
2005/0165909 A1 7/2005 Cromer et al.
2005/0170776 A1 8/2005 Siorpaes
2005/0183143 A1 8/2005 Anderholm et al.
2005/0188038 A1 8/2005 Yabe
2005/0188048 A1 8/2005 Yuan et al.
2005/0193036 A1 9/2005 Phillips et al.
2005/0193096 A1 9/2005 Yu et al.
2005/0198170 A1 9/2005 LaMay et al.
2005/0203966 A1 9/2005 Labrou et al.
2005/0210104 A1 9/2005 Torvinen
2005/0210125 A1 9/2005 Li
2005/0210150 A1* 9/2005 Bahl 709/245
2005/0213511 A1 9/2005 Reece et al.
2005/0216295 A1 9/2005 Abrahamsohn
2005/0222891 A1 10/2005 Chan et al.
2005/0228812 A1 10/2005 Hansmann et al.
2005/0232295 A1 10/2005 Young
2005/0234860 A1 10/2005 Roever et al.
2005/0235214 A1 10/2005 Shimizu et al.
2005/0246139 A1 11/2005 Rivenbark et al.
2005/0248526 A1 11/2005 Twerdahl et al.
2005/0250452 A1 11/2005 Walton et al.
2005/0251555 A1 11/2005 Little, II
2005/0254443 A1 11/2005 Campbell et al.
2005/0256880 A1 11/2005 Nam Koong et al.
2005/0257258 A1* 11/2005 Kinoshita et al. 726/12
2005/0262220 A1 11/2005 Ecklund et al.
2005/0273804 A1 12/2005 Preisman
2005/0278307 A1 12/2005 Battagin et al.
2005/0278641 A1 12/2005 Mansour et al.
2005/0278647 A1 12/2005 Leavitt et al.
2005/0288006 A1 12/2005 Apfel
2006/0012672 A1 1/2006 Schrader et al.
2006/0020525 A1 1/2006 Borelli et al.
2006/0020580 A1 1/2006 Dettinger et al.
2006/0020804 A1 1/2006 Schleifer et al.
2006/0020947 A1 1/2006 Hallamaa et al.
2006/0021023 A1 1/2006 Stewart et al.
2006/0022048 A1 2/2006 Johnson
2006/0026580 A1 2/2006 Cabillic et al.
2006/0029062 A1 2/2006 Rao et al.
2006/0029063 A1 2/2006 Rao et al.
2006/0029064 A1 2/2006 Rao et al.
2006/0031114 A1 2/2006 Zommers
2006/0031300 A1 2/2006 Kock et al.
2006/0031365 A1 2/2006 Kay et al.
2006/0031428 A1 2/2006 Wikman
2006/0031785 A1 2/2006 Raciborski
2006/0031938 A1 2/2006 Choi
2006/0037071 A1 2/2006 Rao et al.
2006/0046686 A1 3/2006 Hawkins et al.
2006/0047844 A1 3/2006 Deng
2006/0048061 A1 3/2006 Forlenza et al.
2006/0052091 A1 3/2006 Onyon et al.
2006/0052137 A1 3/2006 Randall et al.
2006/0059495 A1 3/2006 Spector
2006/0063544 A1 3/2006 Zhao et al.
2006/0065716 A1 3/2006 Peters
2006/0069686 A1 3/2006 Beyda et al.
2006/0069687 A1 3/2006 Cui et al.
2006/0069715 A1 3/2006 Vayssiere
2006/0069742 A1 3/2006 Segre
2006/0069746 A1 3/2006 Davis et al.
2006/0073810 A1 4/2006 Pyhalammi et al.
2006/0074951 A1 4/2006 Beier et al.
2006/0075028 A1 4/2006 Zager et al.
2006/0075138 A1* 4/2006 Borella H04L 29/12066 709/245
2006/0084410 A1 4/2006 Sutaria et al.
2006/0085503 A1 4/2006 Stoye et al.
2006/0093026 A1 5/2006 Montojo et al.
2006/0093135 A1 5/2006 Fiatal et al.
2006/0099969 A1 5/2006 Staton et al.
2006/0099970 A1 5/2006 Morgan et al.
2006/0112177 A1 5/2006 Barkley et al.
2006/0123042 A1 6/2006 Xie et al.
2006/0129665 A1* 6/2006 Toebes H04L 29/12066 709/223
2006/0132495 A1 6/2006 Anderson
2006/0141962 A1 6/2006 Forbes et al.
2006/0143464 A1 6/2006 Ananthanarayanan et al.
2006/0149591 A1 7/2006 Hauf et al.
2006/0149843 A1 7/2006 Rhoads et al.
2006/0149970 A1 7/2006 Imazu
2006/0155822 A1 7/2006 Yang et al.
2006/0161621 A1 7/2006 Rosenberg
2006/0165226 A1 7/2006 Ernst et al.
2006/0166663 A1 7/2006 Haehnichen et al.
2006/0167969 A1 7/2006 Andreev et al.
2006/0168043 A1 7/2006 Eisenberger et al.
2006/0168164 A1 7/2006 Lemson
2006/0179410 A1 8/2006 Deeds
2006/0184591 A1 8/2006 Backholm et al.
2006/0188864 A1 8/2006 Shah
2006/0190428 A1 8/2006 Jung et al.
2006/0190569 A1 8/2006 Neil et al.
2006/0190984 A1 8/2006 Heard et al.
2006/0192014 A1 8/2006 Hamilton et al.
2006/0195570 A1 8/2006 Zellner et al.
2006/0203765 A1 9/2006 Laroia et al.
2006/0209842 A1 9/2006 Creamer et al.
2006/0212531 A1 9/2006 Kikkawa et al.
2006/0224629 A1 10/2006 Alexander et al.
2006/0230394 A1 10/2006 Forth et al.
2006/0234630 A1 10/2006 Lai
2006/0240804 A1 10/2006 Backholm et al.
2006/0240805 A1 10/2006 Backholm et al.
2006/0242137 A1 10/2006 Shah et al.
2006/0242210 A1 10/2006 Ring et al.
2006/0242320 A1 10/2006 Nettle et al.
2006/0242607 A1 10/2006 Hudson
2006/0252435 A1 11/2006 Henderson et al.
2006/0253456 A1 11/2006 Pacholec et al.
2006/0253575 A1 11/2006 Carter et al.
2006/0253605 A1 11/2006 Sundarrajan et al.
2006/0259517 A1 11/2006 Urscheler et al.
2006/0259923 A1 11/2006 Chiu
2006/0265595 A1 11/2006 Scottodiluzio
2006/0271641 A1* 11/2006 Stavrakos G06F 17/30902 709/217
2006/0271884 A1 11/2006 Hurst
2006/0277265 A1 12/2006 Backholm et al.
2006/0277271 A1 12/2006 Morse et al.
2006/0294071 A1 12/2006 Weare et al.
2006/0294223 A1 12/2006 Glasgow et al.
2006/0294388 A1 12/2006 Abraham et al.
2007/0005738 A1 1/2007 Alexion-Tiernan et al.
2007/0006317 A1 1/2007 Asami et al.
2007/0011367 A1 1/2007 Scott et al.
2007/0019610 A1 1/2007 Backholm et al.
2007/0021065 A1 1/2007 Sengupta et al.
2007/0022118 A1 1/2007 Layne
2007/0027775 A1 2/2007 Hwang

(56) **References Cited**

U.S. PATENT DOCUMENTS

2007/0027832 A1 2/2007 Fiatal et al.
2007/0027886 A1 2/2007 Gent et al.
2007/0027917 A1 2/2007 Ariel et al.
2007/0027920 A1 2/2007 Alvarado et al.
2007/0027921 A1 2/2007 Alvarado et al.
2007/0027930 A1 2/2007 Alvarado et al.
2007/0033531 A1 2/2007 Marsh
2007/0038567 A1 2/2007 Allaire et al.
2007/0038931 A1 2/2007 Allaire et al.
2007/0039049 A1 2/2007 Kupferman et al.
2007/0041393 A1* 2/2007 Westhead et al. 370/428
2007/0044041 A1 2/2007 Beynon et al.
2007/0049258 A1 3/2007 Thibeault
2007/0050591 A1 3/2007 Boyd et al.
2007/0053345 A1 3/2007 Hsu et al.
2007/0060196 A1 3/2007 Sharma
2007/0061393 A1 3/2007 Moore
2007/0067147 A1 3/2007 Huang
2007/0067381 A1 3/2007 Grant et al.
2007/0067424 A1 3/2007 Raciborski et al.
2007/0070931 A1 3/2007 Lewis et al.
2007/0072617 A1 3/2007 Lewis et al.
2007/0076264 A1 4/2007 Pierce et al.
2007/0077949 A1 4/2007 Henderson et al.
2007/0078857 A1 4/2007 Punaganti et al.
2007/0078964 A1 4/2007 East et al.
2007/0088852 A1 4/2007 Levkovitz
2007/0100650 A1 5/2007 Ramer et al.
2007/0101061 A1 5/2007 Baskaran et al.
2007/0105627 A1 5/2007 Campbell
2007/0111764 A1 5/2007 Park et al.
2007/0116223 A1 5/2007 Burke et al.
2007/0118620 A1 5/2007 Cartmell et al.
2007/0123214 A1 5/2007 Mock
2007/0130108 A1 6/2007 Simpson et al.
2007/0130217 A1 6/2007 Linyard et al.
2007/0136533 A1 6/2007 Church et al.
2007/0140193 A1 6/2007 Dosa et al.
2007/0147317 A1 6/2007 Smith et al.
2007/0147411 A1 6/2007 Bijwaard et al.
2007/0150599 A1 6/2007 Neogi et al.
2007/0150881 A1 6/2007 Khawand et al.
2007/0153798 A1 7/2007 Krstulich
2007/0156824 A1 7/2007 Thompson
2007/0156842 A1 7/2007 Vermeulen et al.
2007/0161411 A1 7/2007 Liao et al.
2007/0162514 A1 7/2007 Civetta et al.
2007/0165516 A1 7/2007 Xu et al.
2007/0167178 A1 7/2007 Al-Harbi
2007/0174420 A1 7/2007 Khusial et al.
2007/0174433 A1 7/2007 Mendez et al.
2007/0174470 A1 7/2007 Burgess et al.
2007/0175998 A1 8/2007 Lev
2007/0192122 A1 8/2007 Routson et al.
2007/0198698 A1 8/2007 Boyd et al.
2007/0202850 A1 8/2007 Pantalone et al.
2007/0220080 A1 9/2007 Humphrey
2007/0220099 A1 9/2007 Di Giorgio et al.
2007/0233855 A1 10/2007 Brown et al.
2007/0237318 A1 10/2007 McGary
2007/0240218 A1 10/2007 Tuvell et al.
2007/0245010 A1 10/2007 Arn et al.
2007/0245409 A1 10/2007 Harris et al.
2007/0249365 A1 10/2007 Jendbro
2007/0250591 A1 10/2007 Milic-Frayling et al.
2007/0254631 A1 11/2007 Spooner
2007/0255848 A1 11/2007 Sewall et al.
2007/0264993 A1 11/2007 Hughes
2007/0267492 A1 11/2007 Maclaine Pont
2007/0276925 A1 11/2007 LaJoie et al.
2007/0276926 A1 11/2007 LaJoie et al.
2007/0277231 A1 11/2007 Medvinsky et al.
2007/0288469 A1 12/2007 Shenfield
2007/0290787 A1 12/2007 Fiatal et al.
2007/0293207 A1 12/2007 Guedalia et al.
2007/0293238 A1 12/2007 Fiatal et al.
2007/0293958 A1 12/2007 Stehle et al.
2007/0294295 A1 12/2007 Finkelstein et al.
2007/0294373 A1 12/2007 Harrison
2007/0294763 A1 12/2007 Udezue et al.
2007/0296701 A1 12/2007 Pope et al.
2007/0299918 A1 12/2007 Roberts
2007/0300273 A1 12/2007 Turner
2008/0001717 A1 1/2008 Fiatal
2008/0005695 A1 1/2008 Ozzie et al.
2008/0008095 A1 1/2008 Gilfix
2008/0009344 A1 1/2008 Graham et al.
2008/0010365 A1* 1/2008 Schneider 709/223
2008/0016236 A1 1/2008 Beverly et al.
2008/0020786 A1 1/2008 Smith et al.
2008/0028073 A1* 1/2008 Trabe et al. 709/225
2008/0032718 A1 2/2008 Suresh
2008/0034031 A1 2/2008 Weisbrot et al.
2008/0037787 A1 2/2008 Boynton et al.
2008/0039032 A1 2/2008 Haumont
2008/0043692 A1 2/2008 Morita
2008/0045253 A1 2/2008 Mousseau et al.
2008/0059308 A1 3/2008 Gerken
2008/0059398 A1 3/2008 Tsutsui
2008/0059582 A1 3/2008 Hartikainen et al.
2008/0061142 A1 3/2008 Howcroft et al.
2008/0062997 A1* 3/2008 Nix 370/395.2
2008/0068519 A1 3/2008 Adler et al.
2008/0072324 A1 3/2008 Repasi et al.
2008/0077506 A1 3/2008 Rampell et al.
2008/0077571 A1 3/2008 Harris et al.
2008/0085719 A1 4/2008 Kuchibhotla et al.
2008/0085724 A1 4/2008 Cormier et al.
2008/0086379 A1 4/2008 Dion et al.
2008/0086556 A1 4/2008 Ramalingam et al.
2008/0086574 A1* 4/2008 Raciborski et al. 709/245
2008/0091773 A1 4/2008 Hameen-Anttila
2008/0096526 A1 4/2008 Miettinen et al.
2008/0098062 A1 4/2008 Balia
2008/0098120 A1 4/2008 Johnson et al.
2008/0103877 A1 5/2008 Gerken
2008/0104666 A1 5/2008 Dillaway
2008/0108298 A1 5/2008 Selen et al.
2008/0114881 A1 5/2008 Lee et al.
2008/0117922 A1 5/2008 Cockrell et al.
2008/0125225 A1 5/2008 Lazaridis et al.
2008/0130663 A1 6/2008 Fridman et al.
2008/0133326 A1 6/2008 Goncalves et al.
2008/0133641 A1 6/2008 Gent et al.
2008/0133708 A1 6/2008 Alvarado et al.
2008/0134292 A1 6/2008 Ariel et al.
2008/0140665 A1 6/2008 Ariel et al.
2008/0140794 A1 6/2008 Rybak
2008/0146257 A1 6/2008 Clegg
2008/0148146 A1 6/2008 Estrada et al.
2008/0150704 A1 6/2008 Igoe
2008/0151817 A1 6/2008 Fitchett et al.
2008/0154870 A1 6/2008 Evermann et al.
2008/0155613 A1 6/2008 Benya et al.
2008/0162724 A1* 7/2008 Hietasarka H04L 29/12066 709/245
2008/0166999 A1 7/2008 Guedalia et al.
2008/0167019 A1 7/2008 Guedalia et al.
2008/0168145 A1 7/2008 Wilson
2008/0172662 A1 7/2008 Harris et al.
2008/0178294 A1 7/2008 Hu et al.
2008/0180228 A1 7/2008 Wakefield et al.
2008/0183800 A1 7/2008 Herzog et al.
2008/0186166 A1 8/2008 Zhou et al.
2008/0189365 A1 8/2008 Narayanaswami et al.
2008/0192820 A1 8/2008 Brooks et al.
2008/0195819 A1 8/2008 Dumont
2008/0198995 A1 8/2008 McGary et al.
2008/0200161 A1 8/2008 Morse et al.
2008/0201225 A1 8/2008 Maharajh et al.
2008/0201362 A1 8/2008 Multer et al.
2008/0201477 A1* 8/2008 Gray H04L 12/24 709/227
2008/0201751 A1 8/2008 Ahmed et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2008/0207182 A1 8/2008 Maharajh et al.
2008/0209491 A1 8/2008 Hasek
2008/0214148 A1 9/2008 Ramer et al.
2008/0214157 A1 9/2008 Ramer et al.
2008/0216094 A1 9/2008 Anderson et al.
2008/0220797 A1 9/2008 Meiby et al.
2008/0221715 A1 9/2008 Krzyzanowski et al.
2008/0222271 A1 9/2008 Spires
2008/0228899 A1* 9/2008 Plamondon H04L 67/2852 709/219
2008/0232290 A1 9/2008 Elzur et al.
2008/0233983 A1 9/2008 Park et al.
2008/0235351 A1 9/2008 Banga et al.
2008/0235383 A1* 9/2008 Schneider 709/229
2008/0242370 A1 10/2008 Lando et al.
2008/0256090 A1 10/2008 Dietterich et al.
2008/0263170 A1 10/2008 Caron et al.
2008/0270379 A1 10/2008 Ramakrishna
2008/0271123 A1 10/2008 Ollis et al.
2008/0273498 A1 11/2008 Jalil et al.
2008/0281798 A1 11/2008 Chatterjee et al.
2008/0288659 A1 11/2008 Hasha et al.
2008/0294769 A1 11/2008 Doi et al.
2008/0298386 A1 12/2008 Fiatal
2008/0299956 A1 12/2008 Bailey et al.
2008/0301231 A1 12/2008 Mehta et al.
2008/0301300 A1 12/2008 Toub
2008/0313282 A1 12/2008 Warila et al.
2008/0320577 A1 12/2008 Larduinat
2009/0006116 A1 1/2009 Baker et al.
2009/0006531 A1* 1/2009 Gillum et al. 709/203
2009/0010204 A1 1/2009 Pratt, Jr. et al.
2009/0010259 A1 1/2009 Sirotkin
2009/0012841 A1 1/2009 Saft et al.
2009/0016526 A1 1/2009 Fiatal et al.
2009/0019105 A1 1/2009 Sebastian
2009/0019141 A1 1/2009 Bush et al.
2009/0019153 A1 1/2009 Sebastian
2009/0019485 A1 1/2009 Ellis et al.
2009/0019532 A1 1/2009 Jacobsen et al.
2009/0024794 A1 1/2009 Iyer et al.
2009/0027222 A1 1/2009 Larsson et al.
2009/0031006 A1 1/2009 Johnson
2009/0049166 A1 2/2009 Roman et al.
2009/0049173 A1 2/2009 Pulito et al.
2009/0049482 A1 2/2009 Auerbach et al.
2009/0052372 A1 2/2009 Durazzo et al.
2009/0054034 A1 2/2009 Backholm et al.
2009/0055353 A1 2/2009 Meema
2009/0059936 A1* 3/2009 Van De Poel et al. 370/401
2009/0059950 A1 3/2009 Gao et al.
2009/0063647 A1 3/2009 Backholm et al.
2009/0064346 A1 3/2009 Larsson et al.
2009/0069019 A1* 3/2009 Hayama et al. 455/445
2009/0070526 A1 3/2009 Tetrick et al.
2009/0075683 A1 3/2009 Backholm et al.
2009/0077263 A1 3/2009 Koganti et al.
2009/0077326 A1 3/2009 Motohashi
2009/0081944 A1 3/2009 Yavuz et al.
2009/0086651 A1 4/2009 Luft et al.
2009/0094317 A1 4/2009 Venkitaraman
2009/0100416 A1 4/2009 Brown et al.
2009/0110179 A1 4/2009 Elsey et al.
2009/0119266 A1 5/2009 Fitzpatrick et al.
2009/0122772 A1 5/2009 Jung
2009/0125523 A1 5/2009 Fitzpatrick et al.
2009/0129301 A1* 5/2009 Belimpasakis 370/310
2009/0138427 A1 5/2009 Kalavade
2009/0144632 A1 6/2009 Mendez
2009/0147008 A1 6/2009 Do et al.
2009/0149203 A1 6/2009 Backholm et al.
2009/0156178 A1 6/2009 Elsey et al.
2009/0157792 A1 6/2009 Fiatal
2009/0164433 A1 6/2009 R et al.
2009/0164560 A1 6/2009 Fiatal
2009/0164605 A1 6/2009 Lusher et al.
2009/0165115 A1 6/2009 Toumura et al.
2009/0172565 A1 7/2009 Jackson et al.
2009/0172801 A1* 7/2009 Friedrich et al. 726/12
2009/0181641 A1 7/2009 Fiatal
2009/0182500 A1 7/2009 Dicke
2009/0187939 A1 7/2009 Lajoie
2009/0191903 A1 7/2009 Fiatal
2009/0193130 A1 7/2009 Fiatal
2009/0193338 A1 7/2009 Fiatal
2009/0204682 A1 8/2009 Jeyaseelan et al.
2009/0215504 A1 8/2009 Lando
2009/0221326 A1 9/2009 Roussel et al.
2009/0222584 A1* 9/2009 Josefsberg et al. 709/245
2009/0225778 A1* 9/2009 Sharif-Ahmadi et al. 370/469
2009/0228545 A1 9/2009 Mendez et al.
2009/0233595 A1 9/2009 Harris et al.
2009/0241180 A1 9/2009 Fiatal
2009/0248670 A1 10/2009 Fiatal
2009/0248696 A1 10/2009 Rowles et al.
2009/0248794 A1 10/2009 Helms et al.
2009/0248878 A1 10/2009 Tran et al.
2009/0249482 A1 10/2009 Sarathy
2009/0252136 A1 10/2009 Mahany et al.
2009/0254589 A1 10/2009 Nair et al.
2009/0254971 A1 10/2009 Herz et al.
2009/0264138 A1 10/2009 Kang et al.
2009/0268672 A1 10/2009 Kline et al.
2009/0271491 A1 10/2009 Pan
2009/0282125 A1 11/2009 Jeide et al.
2009/0282130 A1 11/2009 Antoniou et al.
2009/0286531 A1 11/2009 Bhatt et al.
2009/0287750 A1 11/2009 Banavar et al.
2009/0299817 A1 12/2009 Fok et al.
2009/0307133 A1 12/2009 Holloway et al.
2009/0307196 A1 12/2009 Shuster
2009/0318171 A1 12/2009 Backholm et al.
2009/0323678 A1 12/2009 Wang
2009/0325565 A1 12/2009 Backholm
2009/0327390 A1 12/2009 Tran et al.
2009/0327819 A1 12/2009 Pan
2010/0010993 A1 1/2010 Hussey, Jr. et al.
2010/0036885 A1 2/2010 Shen et al.
2010/0042691 A1 2/2010 Maguire
2010/0042718 A1 2/2010 Morris
2010/0049852 A1 2/2010 Whitnah et al.
2010/0049872 A1 2/2010 Roskind
2010/0057924 A1 3/2010 Rauber et al.
2010/0067413 A1 3/2010 Schneider et al.
2010/0069127 A1 3/2010 Fiennes
2010/0077035 A1 3/2010 Li et al.
2010/0077083 A1 3/2010 Tran et al.
2010/0082811 A1 4/2010 Van Der Merwe et al.
2010/0083255 A1 4/2010 Bane et al.
2010/0087167 A1 4/2010 Tsurutome et al.
2010/0088722 A1 4/2010 Jiang
2010/0093273 A1 4/2010 Hohl
2010/0095018 A1 4/2010 Khemani et al.
2010/0095065 A1 4/2010 Gray et al.
2010/0099421 A1 4/2010 Patel et al.
2010/0100952 A1 4/2010 Sample et al.
2010/0115050 A1 5/2010 Sultenfuss et al.
2010/0118190 A1 5/2010 Salfati et al.
2010/0121744 A1 5/2010 Belz et al.
2010/0131593 A1 5/2010 Kihara et al.
2010/0131617 A1 5/2010 Osborne et al.
2010/0146107 A1 6/2010 Fiatal
2010/0146607 A1 6/2010 Piepenbrink et al.
2010/0149975 A1 6/2010 Tripathi et al.
2010/0154044 A1 6/2010 Manku
2010/0174735 A1 7/2010 Fiatal
2010/0174756 A1 7/2010 Lazaridis et al.
2010/0174939 A1 7/2010 Vexler
2010/0180005 A1 7/2010 Sebastian et al.
2010/0186011 A1 7/2010 Magenheimer
2010/0192212 A1 7/2010 Raleigh
2010/0197282 A1 8/2010 Uchida
2010/0198752 A1 8/2010 Digon et al.
2010/0203876 A1 8/2010 Krishnaswamy

(56) **References Cited**

U.S. PATENT DOCUMENTS

2010/0207870	A1	8/2010	Cho
2010/0211651	A1	8/2010	Guedalia et al.
2010/0212010	A1	8/2010	Stringer et al.
2010/0214984	A1	8/2010	Cho et al.
2010/0223364	A1	9/2010	Wei
2010/0227594	A1	9/2010	DeVries
2010/0228863	A1	9/2010	Kawauchi
2010/0229096	A1	9/2010	Maiocco et al.
2010/0235473	A1	9/2010	Koren et al.
2010/0238915	A1	9/2010	Cayla et al.
2010/0250706	A1	9/2010	Burckart et al.
2010/0250733	A1	9/2010	Turanyi et al.
2010/0250986	A1	9/2010	Black et al.
2010/0251366	A1	9/2010	Baldry
2010/0257580	A1	10/2010	Zhao
2010/0260038	A1	10/2010	Dhodapkar et al.
2010/0262487	A1	10/2010	Edwards et al.
2010/0262664	A1	10/2010	Brown et al.
2010/0268757	A1	10/2010	Fisher
2010/0274983	A1	10/2010	Murphy et al.
2010/0279662	A1	11/2010	Kuusinen et al.
2010/0281112	A1	11/2010	Plamondon
2010/0284327	A1	11/2010	Miklos
2010/0293335	A1	11/2010	Muthiah et al.
2010/0299223	A1	11/2010	Crouch
2010/0299455	A1	11/2010	Master et al.
2010/0299518	A1	11/2010	Viswanathan et al.
2010/0313018	A1	12/2010	Jorgensen
2010/0315535	A1	12/2010	Nurit et al.
2010/0317340	A1	12/2010	Lee et al.
2010/0319054	A1	12/2010	Mehta et al.
2010/0322124	A1	12/2010	Luoma et al.
2010/0323664	A1	12/2010	Sivaram et al.
2010/0325306	A1	12/2010	Vimpari et al.
2010/0332680	A1	12/2010	Anderson
2011/0019978	A1	1/2011	Jagmag
2011/0022812	A1	1/2011	van der Linden et al.
2011/0028129	A1	2/2011	Hutchison et al.
2011/0029378	A1	2/2011	Ramer et al.
2011/0035799	A1	2/2011	Handler
2011/0040718	A1	2/2011	Tendjoukian et al.
2011/0044304	A1	2/2011	Connelly et al.
2011/0065419	A1	3/2011	Book et al.
2011/0065424	A1	3/2011	Estevez et al.
2011/0066646	A1	3/2011	Danado et al.
2011/0066715	A1	3/2011	Schieder et al.
2011/0072306	A1	3/2011	Racey et al.
2011/0083186	A1	4/2011	Niemela et al.
2011/0093725	A1	4/2011	Theocharous et al.
2011/0093917	A1	4/2011	Alcorn et al.
2011/0095903	A1	4/2011	Gudlavenkatasiva et al.
2011/0099363	A1	4/2011	Boynton et al.
2011/0113109	A1	5/2011	LeVasseur et al.
2011/0119134	A1	5/2011	Zivkovic et al.
2011/0119217	A1	5/2011	Moon et al.
2011/0126060	A1	5/2011	Grube et al.
2011/0126250	A1	5/2011	Turner
2011/0138102	A1	6/2011	Glikson et al.
2011/0138402	A1	6/2011	Fleming
2011/0145646	A1	6/2011	Harris et al.
2011/0151944	A1	6/2011	Morgan
2011/0153816	A1	6/2011	Lloyd et al.
2011/0153937	A1	6/2011	Annamalaisami et al.
2011/0158239	A1	6/2011	Mohaban
2011/0161484	A1	6/2011	Van den Bogaert et al.
2011/0165889	A1	7/2011	Fiatal et al.
2011/0170464	A1	7/2011	Sengottaiyan et al.
2011/0173055	A1	7/2011	Ross et al.
2011/0177847	A1	7/2011	Huang
2011/0179138	A1	7/2011	Van Geest et al.
2011/0179377	A1	7/2011	Fleming
2011/0182220	A1	7/2011	Black et al.
2011/0184827	A1	7/2011	Hubert
2011/0185355	A1	7/2011	Chawla et al.
2011/0189997	A1	8/2011	Tiwari et al.
2011/0190014	A1	8/2011	Fiatal
2011/0191474	A1	8/2011	Fiatal
2011/0201304	A1	8/2011	Sutaria et al.
2011/0207436	A1	8/2011	van Gent et al.
2011/0208810	A1	8/2011	Li et al.
2011/0213800	A1	9/2011	Saros et al.
2011/0213898	A1	9/2011	Fiatal et al.
2011/0214182	A1	9/2011	Adams et al.
2011/0219133	A1	9/2011	Shanmugham
2011/0225646	A1	9/2011	Crawford
2011/0237222	A1	9/2011	Niejadlik
2011/0238772	A1	9/2011	Fiatal
2011/0246950	A1	10/2011	Luna et al.
2011/0252088	A1	10/2011	Fiatal
2011/0264622	A1	10/2011	Vargas et al.
2011/0264731	A1	10/2011	Knowles et al.
2011/0265174	A1	10/2011	Thornton et al.
2011/0294463	A1	12/2011	Fiatal
2011/0294464	A1	12/2011	Fiatal
2011/0296050	A1	12/2011	Cherukuri
2011/0296120	A1	12/2011	Khan
2011/0296415	A1	12/2011	Khan et al.
2011/0302154	A1	12/2011	Snyder
2011/0307541	A1	12/2011	Walsh et al.
2012/0005276	A1	1/2012	Guo et al.
2012/0008536	A1	1/2012	Tervahauta et al.
2012/0022980	A1	1/2012	Angelone
2012/0023190	A1	1/2012	Backholm et al.
2012/0023226	A1	1/2012	Petersen et al.
2012/0023236	A1	1/2012	Backholm et al.
2012/0030280	A1	2/2012	Wang et al.
2012/0030750	A1	2/2012	Bargava et al.
2012/0054386	A1	3/2012	Hanes
2012/0072910	A1	3/2012	Martin et al.
2012/0077482	A1	3/2012	Backholm
2012/0078725	A1	3/2012	Maitra et al.
2012/0078996	A1	3/2012	Shah
2012/0084397	A1	4/2012	Shinohara
2012/0096058	A1	4/2012	Mameri et al.
2012/0096092	A1	4/2012	Davidge et al.
2012/0099592	A1	4/2012	Ludwig
2012/0108200	A1	5/2012	Rubin et al.
2012/0108225	A1	5/2012	Luna et al.
2012/0110109	A1	5/2012	Luna et al.
2012/0110110	A1	5/2012	Luna et al.
2012/0110111	A1	5/2012	Luna et al.
2012/0110112	A1	5/2012	Luna et al.
2012/0110118	A1	5/2012	Luna et al.
2012/0110171	A1	5/2012	Luna et al.
2012/0110173	A1	5/2012	Luna et al.
2012/0110174	A1	5/2012	Wootton et al.
2012/0110275	A1	5/2012	Ganti et al.
2012/0130973	A1	5/2012	Tamm et al.
2012/0131095	A1	5/2012	Luna et al.
2012/0131184	A1	5/2012	Luna et al.
2012/0135726	A1	5/2012	Luna et al.
2012/0140750	A1	6/2012	Yan et al.
2012/0144038	A1	6/2012	Hildebrand
2012/0149352	A1	6/2012	Backholm et al.
2012/0151044	A1	6/2012	Luna et al.
2012/0157170	A1	6/2012	Backholm et al.
2012/0158837	A1	6/2012	Kaul
2012/0158908	A1	6/2012	Luna et al.
2012/0170496	A1	7/2012	Yang et al.
2012/0170569	A1	7/2012	Al-Khudairi
2012/0173616	A1	7/2012	Luna et al.
2012/0174220	A1	7/2012	Rodriguez
2012/0176968	A1	7/2012	Luna
2012/0178414	A1	7/2012	Fiatal
2012/0185597	A1	7/2012	Luna
2012/0185918	A1	7/2012	Backholm et al.
2012/0198046	A1	8/2012	Shah et al.
2012/0209923	A1	8/2012	Mathur et al.
2012/0210121	A1	8/2012	Boynton et al.
2012/0226767	A1	9/2012	Luna et al.
2012/0227059	A1	9/2012	Fleming
2012/0246333	A1	9/2012	Fiatal
2012/0254417	A1	10/2012	Luna
2012/0265836	A1	10/2012	Nemoto et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2012/0271903	A1	10/2012	Luna
2012/0271908	A1	10/2012	Luna et al.
2012/0278431	A1	11/2012	Luna
2012/0278432	A1	11/2012	Luna
2012/0278464	A1	11/2012	Lehane et al.
2012/0278886	A1	11/2012	Luna
2012/0284356	A1	11/2012	Luna
2012/0289239	A1	11/2012	Luna et al.
2012/0290642	A1	11/2012	Shaughnessy et al.
2012/0290675	A1	11/2012	Luna et al.
2012/0290717	A1	11/2012	Luna
2012/0304288	A1	11/2012	Wright et al.
2012/0317284	A1	12/2012	Raleigh et al.
2012/0317370	A1	12/2012	Luna
2012/0331059	A1	12/2012	Luna
2012/0331087	A1	12/2012	Luna et al.
2013/0010693	A1	1/2013	Luna et al.
2013/0012180	A1	1/2013	Backholm et al.
2013/0023232	A1	1/2013	Mendiola
2013/0031191	A1	1/2013	Bott
2013/0031599	A1	1/2013	Luna et al.
2013/0031600	A1	1/2013	Luna et al.
2013/0031601	A1	1/2013	Bott
2013/0041974	A1	2/2013	Luna et al.
2013/0110636	A1	5/2013	Bott
2013/0110637	A1	5/2013	Bott
2013/0142050	A1	6/2013	Luna
2013/0145010	A1	6/2013	Luna et al.
2013/0145017	A1	6/2013	Luna
2013/0151648	A1	6/2013	Luna
2013/0151649	A1	6/2013	Luna
2013/0151709	A1	6/2013	Luna
2013/0159395	A1	6/2013	Backholm et al.
2013/0159511	A1	6/2013	Backholm et al.
2013/0166669	A1	6/2013	Luna et al.
2013/0170348	A1	7/2013	Luna et al.
2013/0173756	A1	7/2013	Luna et al.
2013/0178195	A1	7/2013	Luna et al.
2013/0182572	A1	7/2013	Backholm et al.
2013/0203433	A1	8/2013	Luna et al.
2013/0205366	A1	8/2013	Luna et al.

FOREIGN PATENT DOCUMENTS

EP	0993165	A2	4/2000
EP	1278390	A1	1/2003
EP	1135741	B1	2/2004
EP	1422899	A1	5/2004
EP	1466261	A1	10/2004
EP	1466435	A1	10/2004
EP	1482702	A1	12/2004
EP	1483689	A1	12/2004
EP	1775911	A1	4/2007
EP	1815634	A1	8/2007
EP	1815652	A1	8/2007
EP	1817883	A1	8/2007
EP	2120014	A1	11/2009
EP	2122973	A1	11/2009
EP	2206390	A1	7/2010
EP	2267968	A1	12/2010
EP	2332294	A1	6/2011
EP	2378712	A1	10/2011
EP	2395412	A1	12/2011
EP	2396953	A1	12/2011
EP	2465275	A2	6/2012
EP	2503473	A1	9/2012
EP	2556441	A2	2/2013
EP	2591628	A1	5/2013
EP	2599003	A1	6/2013
EP	2599004	A1	6/2013
EP	2599280	A2	6/2013
EP	2599345	A2	6/2013
EP	2599346	A2	6/2013
EP	2599363	A2	6/2013
EP	2621144	A1	7/2013
FI	117152	B	6/2006
FI	118288	B	9/2007
FI	119581	B	12/2008
FI	123227	B	12/2013
GB	2415335	A	12/2005
GB	2476354	A1	6/2011
GB	2493473	A	2/2013
GB	2495058	A	3/2013
GB	2495066	A	3/2013
GB	2495263	A	4/2013
GB	2495455	A	4/2013
GB	2495463	A	4/2013
GB	2495877	A	4/2013
GB	2496537	A1	5/2013
GB	2497012	A1	5/2013
GB	2493473	B	6/2013
GB	2498064	A	7/2013
GB	2499089	A	8/2013
GB	2499306	A	8/2013
JP	4154233	A	5/1992
JP	10-336372	A	12/1998
JP	2001-218185	A	8/2001
JP	2001-350718	A	12/2001
JP	2001-356973	A	12/2001
JP	2005-515664	T	5/2005
JP	2009-207177	A	9/2009
JP	4386732	B2	10/2009
KR	2001-0018568	A	3/2001
KR	2006-0068186	A	6/2006
KR	2007-0071858	A1	7/2007
KR	10-0765238	B1	10/2007
KR	2007-0102091	A1	10/2007
KR	2007-0112412	A	11/2007
KR	2007-0117874	A	12/2007
KR	2008-0067477	A	7/2008
KR	2009-0038217	A	4/2009
KR	2009-0054528	A	6/2009
KR	2009-0077515	A	7/2009
KR	2010-0064605	A	6/2010
KR	2011-0138122	A	12/2011
KR	10-1227769	B1	1/2013
KR	10-1227821	B1	1/2013
WO	WO 97/41661	A2	11/1997
WO	WO 98/24257	A1	6/1998
WO	WO 98/58322	A2	12/1998
WO	WO 01/30130	A2	5/2001
WO	WO 02/01836	A2	1/2002
WO	WO 03/007570	A1	1/2003
WO	WO 03/058483	A1	7/2003
WO	WO 03/058879	A1	7/2003
WO	WO 03/065701	A1	8/2003
WO	WO 03/098890	A1	11/2003
WO	WO 2004/017591	A2	2/2004
WO	WO 2004/045171	A1	5/2004
WO	WO 2004/047409	A1	6/2004
WO	WO 2005/015925	A2	2/2005
WO	WO 2005/020108	A1	3/2005
WO	WO 2006/045005	A1	4/2006
WO	WO 2006/045102	A2	4/2006
WO	WO 2006/053952	A1	5/2006
WO	WO 2006/053954	A1	5/2006
WO	WO 2006/058967	A1	6/2006
WO	WO 2007/009252	A1	1/2007
WO	WO 2007/015725	A2	2/2007
WO	WO 2007/015726	A1	2/2007
WO	WO 2007/069245	A2	6/2007
WO	WO 2007/073422	A1	6/2007
WO	WO 2007/127878	A1	11/2007
WO	WO 2007/131914	A1	11/2007
WO	WO 2007/149526	A2	12/2007
WO	WO 2007/149540	A2	12/2007
WO	WO 2008/061042	A2	5/2008
WO	WO 2009/017712	A1	2/2009
WO	WO 2009/132700	A1	11/2009
WO	WO 2009/135290	A1	11/2009
WO	WO 2010/035108	A1	4/2010
WO	WO 2010/068842	A1	6/2010
WO	WO 2010/088074	A1	8/2010

(56) **References Cited**

FOREIGN PATENT DOCUMENTS

WO	WO 2011/126889	A2	10/2011
WO	WO 2012/012109	A2	1/2012
WO	WO 2012/018430	A1	2/2012
WO	WO 2012/018431	A1	2/2012
WO	WO 2012/018477	A2	2/2012
WO	WO 2012/018479	A2	2/2012
WO	WO 2012/018556	A2	2/2012
WO	WO 2012/024030	A2	2/2012
WO	WO 2012/033593	A1	3/2012
WO	WO 2012/051044	A1	4/2012
WO	WO 2012/060995	A2	5/2012
WO	WO 2012/060996	A2	5/2012
WO	WO 2012/060997	A2	5/2012
WO	WO 2012/061430	A2	5/2012
WO	WO 2012/061433	A2	5/2012
WO	WO 2012/061437	A1	5/2012
WO	WO 2012/071283	A1	5/2012
WO	WO 2012/071384	A2	5/2012
WO	WO 2012/094675	A2	7/2012
WO	WO 2012/117157	A1	9/2012
WO	WO 2012/145533	A2	10/2012
WO	WO 2012/145541	A2	10/2012
WO	WO 2012/149216	A2	11/2012
WO	WO 2012/149434	A2	11/2012
WO	WO 2012/161751	A1	11/2012
WO	WO 2013/015835	A1	1/2013
WO	WO 2013/015994	A1	1/2013
WO	WO 2013/015995	A1	1/2013
WO	WO 2013/016663	A2	1/2013
WO	WO 2013/016666	A2	1/2013
WO	WO 2013/055413	A1	4/2013
WO	WO 2013/066464	A1	5/2013
WO	WO 2013/066465	A1	5/2013
WO	WO 2013/085590	A1	6/2013
WO	WO 2013/085591	A1	6/2013
WO	WO 2013/086214	A1	6/2013
WO	WO 2013/086225	A1	6/2013
WO	WO 2013/086447	A1	6/2013
WO	WO 2013/086455	A1	6/2013
WO	WO 2013/090212	A1	6/2013
WO	WO 2013/090821	A1	6/2013
WO	WO 2013/090834	A1	6/2013
WO	WO 2013/103988	A1	7/2013
WO	WO 2013/116852	A1	8/2013
WO	WO 2013/116856	A1	8/2013

OTHER PUBLICATIONS

Reaz et al., "Performance of DNS as Location Manager for Wireless Systems in IP Networks", 2005.*
Meriiam Webster, "unilateral", 2015.*
Merriam-Webster online dictionary, "invalid", 2015.*
Merriam-Webster online dictionary, "invalidate", 2015.*
Merriam-Webster online dictionary, "change" 2015.*
Merriam-Webster online dictionary, "independent", 2015.*
Gulbrandsen et al., "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, 2000.*
Fielding et al., "Hypertext Transfer Protocol—HTTP/1.1", RFC 2616, 1999.*
Perkins, "IP Mobility Support", RFC 2002, 1996.*
Carpenter et al., "Renumbering Needs Work", RFC 1900, 1996.*
Berners-Lee et al., "Hypertext Transfer Protocol—HTTP/1.0", RFC 1945, 1996.*
Mockapetris, "Domain Names—Implementation and Specification", RFC 1035, 1987.*
Mockapetris, "Domain Names—Concepts and Facilities", RFC 1034, 1987.*
Perkins et al., "Using DHCP with computers that move", 1995.*
Allchin, James Edward, "An Architecture for Reliable Decentralized Systems," Ph.D. Thesis, Georgia Institute of Technology, 185 pages, Sep. 1983.
Android Developers, "Date," 10 pages, Oct. 27, 2011.
Augun, Audrey, "Integrating Lotus Notes With Enterprise Data," Lotus Notes Advisory, pp. 22-25, Jul.-Aug. 1996.
Balaban, Bob, "This Is Not Your Father's Basic: LotusScript in Notes Release 4," The View, vol. 1, Issue 5, 32 pages, Nov.-Dec. 1995.
Bedell, Doug, "Meeting Your New Best Friends Six Degrees Widens Your Contacts in Exchange for Sampling Web Sites," The Dallas Morning News, 4 pages, Oct. 27, 1998.
Bergman, Lawrence D. et al., "Programming-By-Demonstration for Behavior-Based User Interface Customization," IBM Research Report, RC23116, 5 pages, Feb. 20, 2004.
B'Far, Reza et al., "Designing Effective User Interfaces for Wireless Devices," Publication Unknown, 14 pages, Published prior to Feb. 23, 2006.
Blaney, Jeff, "You Can Take It With You—An Introduction to Mobile Computing With Notes R4," The View, vol. 2, Issue 1, 14 pages, Jan.-Feb. 1996.
Braden, R., "Requirements for Internet Hosts—Application and Support," RFC 1123, 80 pages, Oct. 1989.
Brown, Kevin et al., "Mastering Lotus Notes®," Sybex Inc., 996 pages, 1995.
"Chapter: About NotesPump," Publication Unknown, 480 pages, Published prior to Jan. 8, 2003.
"Chapter 13-1—Anatomy of a Note ID," Publication Unknown, 8 pages, Published prior to Jan. 8, 2003.
Cole, Barb et al., "Lotus Airs Notes-To-Database Integration Tool," Network World, 2 pages, Oct. 2, 1995.
"CR 3483 to Release 8 TS 25.331, Rev. 2," 3GPP TSG-RAN2 Meeting #64, Prague, Czech Republic, 11 pages, Nov. 10-14, 2008.
"CR 4100 to Release 8 TS 25.331, Rev. 1," 3GPP TSG-RAN WG2 Meeting #69, San Francisco, U.S., 6 pages, Feb. 22-26, 2010.
Dahl, Andrew, "Lotus Notes® 4 Administrator's Survival Guide," Sams Publishing, 64 pages, 1996.
Decker, Stefan et al., "The Social Semantic Desktop," Digital Enterprise Research Institute, Deri Technical Report May 2, 2004, 7 pages, May 2004.
Elz, R. et al., "Clarifications to the DNS Specification," RFC 2181, 12 pages, Jul. 1997.
European Patent Application No. EP 03705704.9, Supplementary European Search Report, 4 pages, Jun. 9, 2010.
European Patent Application No. EP 03707338.4, Supplementary European Search Report, 2 pages, Apr. 18, 2011.
European Patent Application No. EP 05815115.0, Supplementary European Search Report, 7 pages, Nov. 17, 2011.
Falkner, Mike, "How to Plan, Develop, and Implement Lotus Notes® in Your Organization," John Wiley & Sons, Inc., 539 pages, 1996.
Freeland, Pat et al., "Lotus Notes 3-3.1 for Dummies™," IDG Books Worldwide, 389 pages, 1994.
Frenkel, Garry, "Pumping for Info: Notes and Database Integration," Network Computing, 10 pages, May 1, 1996.
Gameline, Advertisement, 1 page, 1982.
Gewirtz, David, "Lotus Notes 3 Revealed!," Prima Publishing, 261 pages, 1994.
Grous, Paul J., "Creating and Managing a Web Site With Lotus Internotes Web Publisher," The View, vol. 1, Issue 4, 20 pages, Sep.-Oct. 1995.
GSM Association, "Network Efficiency Task Force Fast Dormancy Best Practices," V1.0, 21 pages, May 26, 2010.
Haas, Zygmunt J. et al., "Mobile-TCP: An Asymmetric Transport Protocol Design for Mobile Systems," IEEE, pp. 1054-1058, 1997.
Haas, Zygmunt J. et al., "The Design and Performance of Mobile TCP for Wireless Networks," Journal of High Speed Networks, vol. 10, pp. 187-207, 2001.
Hajdu, Kalman et al., "Lotus Notes Release 4 in a Multiplatform Environment," IBM Corporation, 173 pages, Feb. 1996.
Hardy, Ed, "Microsoft Proposes Two New Thumb-Driven User Interfaces," Brighthand Consulting, Inc., 2 pages, 2003.
IBM Corporation, "The Architecture of Lotus Notes," White Paper No. 114654, 26 pages, May 31, 1995.
IBM Corporation, "The History of Notes and Domino," Lotus Developer Domain, 11 pages, Sep. 29, 2003.
ImTOO, "ImTOO iPod Movie Converter," 3 pages, Nov. 9, 2005.
IntelliLink Corporation, "IntelliLink® for Windows User's Guide," Version 3.0, 167 pages, 1994.

(56) **References Cited**

OTHER PUBLICATIONS

International Application No. PCT/US2003/000618, International Search Report, 1 page, Apr. 4, 2003.
International Application No. PCT/US2003/000624, International Search Report, 2 pages, May 13, 2003.
International Application No. PCT/US2005/037702, International Preliminary Examination Report, 6 pages, Nov. 20, 2007.
International Application No. PCT/US2005/037702, International Search Report, 1 page, Nov. 5, 2007.
International Application No. PCT/US2005/037702, Written Opinion, 6 pages, Nov. 5, 2007.
International Application No. PCT/US2005/038135, International Search Report, 2 pages, Aug. 8, 2008.
International Application No. PCT/US2005/038135, Written Opinion, 8 pages, Aug. 8, 2008.
International Application No. PCT/US2005/038135, International Preliminary Report on Patentability, 9 pages, Oct. 31, 2011.
International Application No. PCT/FI2005/050424, International Search Report, 4 pages, Mar. 2, 2006.
International Application No. PCT/FI2005/050426, International Search Report, 3 pages, Mar. 1, 2006.
International Application No. PCT/FI2005/050441, International Search Report, 3 pages, Mar. 1, 2006.
International Application No. PCT/US2006/023426, International Search Report, 1 page, Feb. 21, 2007.
International Application No. PCT/US2006/023427, International Search Report, 1 page, Oct. 12, 2006.
International Application No. PCT/US2007/014462, International Search Report, 1 page, Jul. 2, 2008.
International Application No. PCT/US2007/014497, International Search Report, 1 page, Aug. 25, 2008.
International Application No. PCT/US2011/030534, International Search Report, 10 pages, Dec. 29, 2011.
International Application No. PCT/US2011/037932, International Search Report, 9 pages, Jan. 2, 2012.
International Application No. PCT/US2011/037943, International Search Report, 11 pages, Jan. 2, 2012.
International Application No. PCT/US2011/043322, International Search Report, 9 pages, Feb. 9, 2012.
International Application No. PCT/US2011/043328, International Search Report, 12 pages, Feb. 27, 2012.
International Application No. PCT/US2011/043409, International Search Report, 11 pages, Feb. 9, 2012.
Japanese Patent Application No. 2003-558726, Office Action, 2 pages, Jun. 10, 2008.
Karlson, Amy K. et al., "AppLens and LaunchTile: Two Designs for One-Handed Thumb Use on Small Devices," Proceedings of CHI 2005, 10 pages, Apr. 2-7, 2005.
Kent, S. et al., "Security Architecture for the Internet Protocol," RFC 2401, The Internet Society, 62 pages, Nov. 1998.
Kleinberg, Jon, "The Small-World Phenomenon: An Algorithmic Perspective," Cornell Computer Science Technical Report 99/1776, 14 pages, Oct. 1999.
Koeppel, Dan, "GUIs Just Want to Have Fun," Wired Magazine, Issue 8.10, 12 pages, Oct. 2000.
Kornblith, Polly Russell, "Lotus Notes Answers: Certified Tech Support," Covers Release 3, McGraw-Hill, Inc., 326 pages, 1994.
Kreisle, Bill, "Teach Yourself . . . Lotus Notes 4," MIS Press, 464 pages, 1996.
Lamb, John P. et al., "Lotus Notes Network Design," McGraw-Hill, 278 pages, 1996.
Londergan, Stephen et al., "Lotus Notes® Release 4 for Dummies®," IDG Books Worldwide, 229 pages, 1996.
Lotus Development Corporation, "Firewall Security Overview and How Firewalls Relate to Lotus Notes," Lotus Notes Knowledge Base, 9 pages, May 22, 1996.
Lotus Development Corporation, "How to Set Up 'Firewall' Protection for a Notes Domain," Lotus Notes Knowledge Base, 2 pages, Nov. 6, 1995.
Lotus Development Corporation, "Lotus Announces Lotus NotesPump 1.0," Lotus Notes Knowledge Base, 6 pages, Oct. 31, 1995.
Lotus Development Corporation, "Lotus Inside Notes—The Architecture of Notes and the Domino Server," 207 pages, 2000.
Lotus Development Corporation, "Lotus NotesPump 1.0 Q & A," Lotus Notes Knowledge Base, 3 pages, Oct. 31, 1995.
Lotus Development Corporation, "Lotus NotesPump: Database Integration for Lotus Notes," Lotus Notes Knowledge Base, 5 pages, Oct. 31, 1995.
Lotus Development Corporation, "Lotus Notes Administration," Release 3.3, 20 pages, 1995.
Lotus Development Corporation, "Lotus Notes Administrator's Guide," Release 4, 499 pages, 1995.
Lotus Development Corporation, "Lotus Notes Administrators Guide—Server for NetWare, OS-2, and Unix," Release 3.1, 509 pages, 1994.
Lotus Development Corporation, "Lotus Notes Administrators Guide—Server for Windows," Release 3.1, 345 pages, 1994.
Lotus Development Corporation, "Lotus Notes Application Developer's Guide," Release 4, 475 pages, 1995.
Lotus Development Corporation, "Lotus Notes Customer Service Application Guide," Release 3.1, 46 pages, 1994.
Lotus Development Corporation, "Lotus Notes Customer Support Guide," 33 pages, Published prior to Jan. 8, 2003.
Lotus Development Corporation, "Lotus Notes Customer Support Guide—North American Guide," Release 4.1, 51 pages, Published prior to Jan. 8, 2003.
Lotus Development Corporation, "Lotus Notes Database Manager's Guide," Release 4, 115 pages, 1995.
Lotus Development Corporation, "Lotus Notes Deployment Guide," Release 4, 104 pages, 1995.
Lotus Development Corporation, "Lotus Notes for Windows, OS-2, and Macintosh," Release 3.3, 89 pages, 1995.
Lotus Development Corporation, "Lotus Notes Getting Started With Application Development," Release 3.1, 151 pages, 1994.
Lotus Development Corporation, "Lotus Notes Install Guide for Servers," Release 4, 68 pages, 1996.
Lotus Development Corporation, "Lotus Notes Install Guide for Workstations," Release 4, 28 pages, 1995.
Lotus Development Corporation, "Lotus Notes Install Guide for Workstations," Release 4.1, 67 pages, 1996.
Lotus Development Corporation, "Lotus Notes Install Guide for Workstations," Release 4.5, 81 pages, 1996.
Lotus Development Corporation, "Lotus Notes Internet Cookbook for Notes Release 3," 21 pages, Jan. 16, 1996.
Lotus Development Corporation, "Lotus Notes Internet Cookbook for Notes Release 4," 35 pages, Feb. 14, 1996.
Lotus Development Corporation, "Lotus Notes Internotes Web Navigator Administrator's Guide," Release 4, 60 pages, 1995.
Lotus Development Corporation, "Lotus Notes Internotes Web Navigator User's Guide," Release 4, 56 pages, 1995.
Lotus Development Corporation, "Lotus Notes Internotes Web Publisher Guide," Release 4, 122 pages, 1996.
Lotus Development Corporation, "Lotus Notes LotusScript Classes for Notes," Release 4, 6 pages, Published prior to Jan. 8, 2003.
Lotus Development Corporation, "Lotus Notes Migration Guide," Release 4, 110 pages, 1996.
Lotus Development Corporation, "Lotus Notes Network Configuration Guide," Release 4.5, 121 pages, 1996.
Lotus Development Corporation, "Lotus Notes Network Driver Documentation," Release 3.1, 100 pages, 1994.
Lotus Development Corporation, "Lotus Notes Programmer's Guide—Part 1," Release 4, 614 pages, 1995.
Lotus Development Corporation, "Lotus Notes Programmer's Guide—Part 2," Release 4, 462 pages, 1995.
Lotus Development Corporation, "Lotus Notes Quick Reference for Application Developers," Release 3, 6 pages, Published prior to Jan. 8, 2003.
Lotus Development Corporation, "Lotus Notes Quick Reference for Macintosh," Release 3, 6 pages, Published prior to Jan. 8, 2003.
Lotus Development Corporation, "Lotus Notes Quick Reference for SmartIcons," Release 3.1, 4 pages, Published prior to Jan. 8, 2003.

(56) **References Cited**

OTHER PUBLICATIONS

Lotus Development Corporation, "Lotus Notes Quick Reference for Windows and Presentation Manager," Release 3, 6 pages, Published prior to Jan. 8, 2003.
Lotus Development Corporation, "Lotus Notes Release Notes," Release 4, 139 pages, 1995.
Lotus Development Corporation, "Lotus Notes Release Notes," Release 4.1, 197 pages, 1996.
Lotus Development Corporation, "Lotus Notes Server for Windows," Release 3.3, 7 pages, 1994.
Lotus Development Corporation, "Lotus Notes Server Up and Running!," Release 4, 13 pages, 1996.
Lotus Development Corporation, "Lotus Notes Site and Systems Planning Guide," Release 3.1, 169 pages, 1994.
Lotus Development Corporation, "Lotus Notes Start Here—Workstation Install for Windows, OS-2 and Macintosh," Release 3.3, 47 pages, 1995.
Lotus Development Corporation, "Lotus Notes Step by Step—A Beginner's Guide to Lotus Notes," Release 4, 179 pages, 1995.
Lotus Development Corporation, "Lotus Notes Step by Step—A Beginner's Guide to Lotus Notes," Release 4.1, 167 pages, 1996.
Lotus Development Corporation, "Lotus Software Agreement," 8 pages, Published prior to Jan. 8, 2003.
Lotus Development Corporation, "What Is the Notes Replicator?," Lotus Notes Knowledge Base, 8 pages, Jul. 5, 1995.
"Lotus Notes Advisor," Advisor Publications Inc. 55 pages, Jun. 1995.
"Lotus Notes Advisor," Advisor Publications Inc. 55 pages, Aug. 1995.
"Lotus Notes Advisor," Advisor Publications Inc. 55 pages, Oct. 1995.
"Lotus Notes Advisor," Advisor Publications Inc. 55 pages, Dec. 1995.
"Lotus Notes Advisor," Advisor Publications Inc. 63 pages, Jan.-Feb. 1996.
"Lotus Notes Advisor," Advisor Publications Inc. 55 pages, Apr. 1996.
"Lotus Notes Advisor," Advisor Publications Inc. 55 pages, Jun. 1996.
"Lotus Notes Advisor," Advisor Publications Inc. 55 pages, Aug. 1996.
"Lotus Notes Advisor," Advisor Publications Inc., 55 pages, Oct. 1996.
"Lotus Notes Advisor," Advisor Publications Inc., 63 pages, Dec. 1996.
"Lotus Notes—Notes Administration Help," Screen Shots, 17 pages, Published prior to Jan. 8, 2003.
MacGregor, Rob et al., "The Domino Defense: Security in Lotus Notes and the Internet," IBM Corporation, 183 pages, Dec. 1997.
Maltz, David A. et al., "MSOCKS: An Architecture for Transport Layer Mobility," IEEE, pp. 1037-1045, 1998.
Marmel, Elaine, "Easy Lotus® Notes Release 4.0," Que Corporation, 237 pages, 1996.
Mason, Luke, "Windows XP: New GUI Design Shows Skin Is in," TechRepublic, 4 pages, Apr. 4, 2001.
McMullen, Melanie, "Network Remote Access and Mobile Computing," Miller Freeman Inc., 226 pages, 1994.
Microsoft, Definition of "Access," Microsoft Computer Dictionary, Fifth Edition, 2 pages, May 1, 2002.
Microsoft, Definition of "Synchronization," Microsoft Computer Dictionary, Fifth Edition, 2 pages, May 1, 2002.
Milgram, Stanley, "The Small-World Problem," Psychology Today, vol. 2, pp. 60-67, 1967.
Miller, Victor S., "Use of Elliptic Curves in Cryptography," Advances in Cryptology—CRYPTO '85 Proceedings, vol. 218, pp. 417-426, 1985.
Mockapetris, P., "Domain Names—Concepts and Facilities," RFC 1034, 43 pages, Nov. 1987.
Mockapetris, P., "Domain Names—Implementation and Specification," RFC 1035, 43 pages, Nov. 1987.
Myers, Brad A. et al., "Extending the Windows Desktop Interface With Connected Handheld Computers," WSS'00 Proceedings of the 4th Conference on USENIX Windows Systems Symposium, vol. 4, 10 pages, 2000.
Myers, Brad A. et al., "User Interfaces That Span Hand-Held and Fixed Devices," CHI'2001 Workshop on Distributed and Disappearing User Interfaces in Ubiquitous Computer, 4 pages, 2001.
National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication 197, 52 pages, Nov. 26, 2001.
National Institute of Standards and Technology, "Secure Hash Standard," Federal Information Processing Standards Publication 180-2, 83 pages, Aug. 1, 2002.
Netscape Communications Corporation, "Netscape Mail Server Administrator's Guide," Version 2.0, 172 pages, 1996.
Netscape Communications Corporation, "Netscape Mail Server Installation Guide," Version 2.0 for Unix, 62 pages, 1996.
Netscape Communications Corporation, "Netscape Mail Server User's Guide," Version 2.0, 35 pages, 1996.
Netscape Communications Corporation, "Netscape News Server Administrator's Guide for Windows NT," Version 2.0, 119 pages, 1996.
Niederee, Claudia et al., "A Multi-Dimensional, Unified User Model for Cross-System Personalization," Proceedings of the AVI 2004 Workshop on Environments for Personalized Information Access, 11 pages, 2004.
Nokia, "Developer Platforms," 3 pages, 2005.
"NotesPump 1.0 Release Notes," Publication Unknown, 8 pages, Published prior to Jan. 8, 2003.
Opyt, Barbara et al., "Use the Internet As Your Lotus Notes WAN," Lotus Notes Advisor, pp. 17-20, Nov.-Dec. 1996.
Ortiz, C. Enrique, "An Introduction to the Symbian OS™ Platform for Palm OS® Developers," Metrowerks Corp., 21 pages, 2002.
"Overview—What Is Lotus NotesPump?," Publication Unknown, 88 pages, Published prior to Jan. 8, 2003.
Perez, Sarah, "Onavo's Data-Compressing Mobile App Raises $10 Million Series B From Horizons, Motorola Ventures," 2 pages, Jan. 24, 2012.
Phillips, Joshua et al., "Modeling the Intelligence Analysis Process for Intelligent User Agent Development," Research and Practice in Human Resource Management, vol. 9, No. 1, pp. 59-73, 2001.
Pyle, Hugh, "The Architecture of Lotus Notes," Lotus Notes Advisor, Premiere Issue, pp. 18-27, 1995.
Pyle, Lisa, "A Jump Start to the Top Ten R3-To-R4 Migration Considerations," The View, vol. 1, Issue 5, 22 pages, Nov.-Dec. 1995.
Qualcomm Incorporated, "Managing Background Data Traffic in Mobile Devices," 16 pages, Jan. 2012.
Qualcomm, "System Parameter Recommendations to Optimize PS Data User Experience and UE Battery Life," 80-W1112-1, Revision B, 9 pages, Mar. 2007.
Ringel, Meredith et al., "iStuff: A Scalable Architecture for Lightweight, Wireless Devices for Ubicomp User Interfaces," Proceedings of UbiComp 2002, 2 pages, 2002.
Shafran, Andrew Bryce, "Easy Lotus Notes® for Windows™," Que Corporation, 199 pages, 1994.
Signorini, Eugene, "SEVEN's Service-Based Wireless Solutions Enable Enterprises to Untether E-Mail," Wireless/Mobile Enterprise & Commerce, 16 pages, Oct. 2004.
Swedeen, Bret et al., "Under the Microscope—Domino Replication," LDD Today, 8 pages, Oct. 1, 1998.
Tamura, Randall A., "Lotus® Notes™ 4 Unleashed," Sams Publishing, 928 pages, 1996.
U.S. Appl. No. 60/663,463, File History, 113 pages, Mar. 18, 2005.
Vivacqua, Adriana et al., "Profiling and Matchmaking Strategies in Support of Opportunistic Collaboration," CoopIS/DOA/ODBASE 2003, LNCS 2888, pp. 162-177, 2003.
Wainwright, Andrew, "Secrets to Running Lotus Notes: The Decisions No One Tells You How to Make," IBM Corporation, 193 pages, Oct. 1996.
Wilcox, Adam A., "PC Learning Labs Teaches Lotus Notes 3.0," Ziff-Davis Press, 381 pages, 1993.

(56) **References Cited**

OTHER PUBLICATIONS

Wong, Harry, "Casahl's Replic-Action: Delivering True Notes—DBMS Integration," The View, vol. 2, Issue 1, pp. 33-50, Jan.-Feb. 1996.
Eronen, "TCP Wake-Up: Reducing Keep-Alive Traffic in Mobile IPv4 and Ipsec NAT Traversal," NRC-TR-2008-002, Nokia, 10 pages, Jan. 31, 2008.
International Application No. PCT/US2011/044974, International Search Report, 15 pages, Jun. 1, 2012.
International Application No. PCT/US2011/056474, International Search Report & Written Opinion, 9 pages, May 4, 2012.
International Application No. PCT/US2011/056476, International Search Report & Written Opinion, 12 pages, May 24, 2012.
International Application No. PCT/US2011/056478, International Search Report & Written Opinion, 11 pages, May 31, 2012.
International Application No. PCT/US2011/058840, International Search Report & Written Opinion, 10 pages, Apr. 26, 2012.
International Application No. PCT/US2011/058843, International Search Report & Written Opinion, 11 pages, May 16, 2012.
International Application No. PCT/US2011/058848, International Search Report & Written Opinion, 10 pages, Apr. 10, 2012.
International Application No. PCT/US2011/061512, International Search Report, 10 pages, May 10, 2012.
International Application No. PCT/US2011/061795, International Search Report & Written Opinion, 10 pages, Jul. 31, 2012.
International Application No. PCT/US2012/021459, International Search Report & Written Opinion, 10 pages, Jun. 1, 2012.
International Application No. PCT/US2012/022121, International Search Report & Written Opinion, 11 pages, May 14, 2012.
Newton, Harry, "Newton's Telecom Dictionary," 20th Edition, pp. 67, 127, 542, Mar. 2004.
Seven Networks, Inc., "Seven Optimizing the Mobile Ecosystem," www.seven.com/products.traffic_optimization.php, 1 page, May 29, 2012.
Wikipedia, Definition for "General Packet Radio Service," 7 pages, downloaded on May 31, 2012.
Adwankar, Sandeep et al., "Universal Manager: Seamless Management of Enterprise Mobile and Non-Mobile Devices," Proceedings of the 2004 IEEE International Conference on Mobile Data Management, 12 pages, 2004.
Amato, Guiseppe et al., "Detection of Images With Adult Content for Parental Control on Mobile Devices," Mobility, 5 pages, 2009.
Ankeny, Jason, "F-Secure: Android to Blame for 79% of All Mobile Malware in 2012," FierceMobileContent, FierceMarkets, 3 pages, Mar. 7, 2013.
Armstrong, Trevor et al., "Efficient and Transparent Dynamic Content Updates for Mobile Clients," The Fourth International Conference on Mobile Systems, Applications and Services, MobiSys 2006, pp. 56-68, Jun. 19-22, 2006.
Baset, Salman et al., "An Analysis of the Skype Peer-To-Peer Internet Telephony Protocol," Columbia University, Department of Computer Science, 12 pages, Sep. 15, 2004.
Blefari-Melazzi, N. et al., "Autonomic Control and Personalization of a Wireless Access Network," Computer Networks, vol. 51, pp. 2645-2676, 2007.
Canadian Patent Application No. 2,806,527, Office Action, 4 pages, Apr. 3, 2013.
Canadian Patent Application No. 2,806,529, Office Action, 4 pages, Apr. 2, 2013.
Canadian Patent Application No. 2,806,549, Office Action, 2 pages, Jul. 23, 2013.
Canadian Patent Application No. 2,806,550, Office Action, 3 pages, May 23, 2013.
Canadian Patent Application No. 2,806,557, Office Action, 4 pages, Mar. 21, 2013.
Canadian Patent Application No. 2,806,557, Office Action, 4 pages, Jul. 18, 2013.
de la Iglesia, Didac Gil et al., "Enhancing Mobile Learning Activities by the Use of Mobile Virtual Devices—Some Design and Implementation Issues," 2010 International Conference on Intelligent Networking and Collaborative Systems, IEEE Computer Society, pp. 137-144, 2010.
European Patent Application No. EP 12775986.8, Supplementary European Search Report, 6 pages, Jul. 2, 2013.
European Patent Application No. EP 13150313.8, Examination Report, 5 pages, Jul. 19, 2013.
European Patent Application No. EP 13150313.8, Supplementary European Search Report, 3 pages, Jul. 2, 2013.
Fukushima, Yukinobu et al., "Planning Method of Robust WDM Networks Against Traffic Changes," IEIC Technical Report, vol. 103, No. 1, pp. 11-16, 2003.
International Application No. PCT/US2012/020669, International Search Report & Written Opinion, 10 pages, Sep. 12, 2012.
International Application No. PCT/US2012/034288, International Search Report & Written Opinion, 15 pages, Nov. 23, 2012.
International Application No. PCT/US2012/034297, International Search Report & Written Opinion, 11 pages, Nov. 26, 2012.
International Application No. PCT/US2012/034300, International Search Report & Written Opinion, 9 pages, Nov. 23, 2012.
International Application No. PCT/US2012/035292, International Search Report & Written Opinion, 11 pages, Nov. 28, 2012.
International Application No. PCT/US2012/035300, International Search Report & Written Opinion, 9 pages, Nov. 28, 2012.
International Application No. PCT/US2012/035608, International Search Report & Written Opinion, 9 pages, Nov. 28, 2012.
International Application No. PCT/US2012/035617, International Search Report & Written Opinion, 9 pages, Oct. 10, 2012.
International Application No. PCT/US2012/042982, International Search Report & Written Opinion, 11 pages, Jan. 2, 2013.
International Application No. PCT/US2012/046317, International Search Report & Written Opinion, 10 pages, Jan. 3, 2013.
International Application No. PCT/US2012/046321, International Search Report & Written Opinion, 11 pages, Dec. 27, 2012.
International Application No. PCT/US2012/048623, International Search Report & Written Opinion, 13 pages, Jan. 31, 2013.
International Application No. PCT/US2012/048639, International Search Report & Written Opinion, 15 pages, Jan. 29, 2013.
International Application No. PCT/US2012/050467, International Search Report & Written Opinion, 14 pages, Mar. 4, 2013.
International Application No. PCT/US2012/050476, International Search Report & Written Opinion, 14 pages, Feb. 28, 2013.
International Application No. PCT/US2012/055931, International Search Report & Written Opinion, 9 pages, Mar. 4, 2013.
International Application No. PCT/US2012/055934, International Search Report & Written Opinion, 12 pages, Jan. 31, 2013.
International Application No. PCT/US2012/068278, International Search Report & Written Opinion, 11 pages, Mar. 21, 2013.
International Application No. PCT/US2012/068291, International Search Report & Written Opinion, 10 pages, Mar. 21, 2013.
International Application No. PCT/US2012/068612, International Search Report & Written Opinion, 10 pages, Mar. 29, 2013.
International Application No. PCT/US2012/068624, International Search Report & Written Opinion, 10 pages, Mar. 25, 2013.
International Application No. PCT/US2012/068822, International Search Report & Written Opinion, 13 pages, Mar. 29, 2013.
International Application No. PCT/US2012/069917, International Search Report & Written Opinion, 14 pages, Apr. 30, 2013.
International Application No. PCT/US2012/069931, International Search Report & Written Opinion, 10 pages, Apr. 30, 2013.
International Application No. PCT/US2013/020574, International Search Report & Written Opinion, 9 pages, Mar. 4, 2013.
International Application No. PCT/US2013/024657, International Search Report & Written Opinion, 14 pages, May 30, 2013.
International Application No. PCT/US2013/024664, International Search Report & Written Opinion, 11 pages, Apr. 1, 2013.
International Application No. PCT/US2013/027694, International Search Report & Written Opinion, 12 pages, Jun. 4, 2013.
International Application No. PCT/US2013/035257, International Search Report & Written Opinion, 14 pages, Jul. 26, 2013.
International Application No. PCT/US2013/036013, International Search Report & Written Opinion, 16 pages, Jul. 26, 2013.

(56) **References Cited**

OTHER PUBLICATIONS

Johnsen, Lotte, Master's Thesis for "Content Distribution in Ad Hoc Networks," Norwegian University of Science and Technology, Department of Telematics, 158 pages, Spring 2006.
Kanter, Theo et al., "Smart Delivery of Multimedia Content for Wireless Applications," Computer Science, vol. 1818, pp. 70-81, 2000.
Kino, Toru, "Infrastructure Technology for Cloud Services," Fujitsu Sci. Tech. J., vol. 47, No. 4, pp. 434-442, Oct. 2011.
LeBrun, Jason et al., "Bluetooth Content Distribution Stations on Public Transit," ACM, Inc., 3 pages, 2006.
Open Mobile Alliance Ltd., "OMA AOI Architecture Principles—OMA-CD-AOI-2012-0012," 12 pages, Dec. 17, 2012.
Openet Telecom, "Taming Signaling: Addressing the Signaling Storm," Openet Labs Technical White Paper, 11 pages, 2012.
Parker, Tammy, "SK Telecom Aims to License, Standardize Smart Push," FierceBroadbandWireless, 4 pages, Aug. 26, 2012.
Paul, Sanjoy et al., "The Cache-And-Forward Network Architecture for Efficient Mobile Content Delivery Services in the Future Internet," First ITU-T Kaleidoscope Academic Conference for Innovations in NGN—Future Network and Services, 7 pages, May 12-13, 2008.
Qualcomm Incorporated, "A 3G/LTE Wi-Fi Offload Framework: Connectivity Engine (CnE) to Manage Inter-System Radio Connections and Applications," 15 pages, Jun. 2011.
Sung, Dan Keun, Ph.D., "EE624 Mobile Communications Systems (MCS)," Korea Advanced Institute of Science and Technology, Department of Electrical Engineering and Computer Science, 13 pages, Fall 2000.
U.K. Patent Application No. GB1219986.5, Examination Report, 6 pages, Dec. 24, 2012.
U.K. Patent Application No. GB1222083.6, Search Report, 4 pages, Apr. 30, 2013.
U.K. Patent Application No. GB1222636.1, Search Report, 4 pages, May 30, 2013.
U.K. Patent Application No. GB1300808.1, Examination Report, 10 pages, Mar. 11, 2013.
U.K. Patent Application No. GB1301235.6, Examination Report, 3 pages, Jun. 24, 2013.
U.K. Patent Application No. GB1301258.8, Examination Report, 5 pages, Feb. 18, 2013.
U.K. Patent Application No. GB1301271.1, Examination Report, 4 pages, Mar. 14, 2013.
U.K. Patent Application No. GB1302158.9, Examination Report, 7 pages, Mar. 15, 2013.
U.K. Patent Application No. GB1302515.0, Examination Report, 3 pages, Mar. 5, 2013.
U.K. Patent Application No. GB1307218.6, Examination Report, 5 pages, Jun. 24, 2013.
U.K. Patent Application No. GB1307573.4, Examination Report, 4 pages, Jul. 3, 2013.
U.K. Patent Application No. GB1309204.4, Examination Report, 6 pages, Jun. 25, 2013.
U.K. Patent Application No. GB1309373.7, Examination Report, 6 pages, Jun. 26, 2013.
U.K. Patent Application No. GB1310340.3, Examination Report, 8 pages, Jul. 10, 2013.
U.K. Patent Application No. GB1310348.6, Examination Report, 5 pages, Jul. 15, 2013.
Yin, Chunjiang et al., "IST-2001-32125 Flows," Deliverable No. D15, Information Society Technologies, 97 pages, Dec. 22, 2003.
Zhang, Qi et al., "Cloud Computing: State-Of-The-Art and Research Challenges," J Internet Serv Appl, vol. 1, pp. 7-18, 2010.
European Patent Office Search and Examination report for Application No. 12732122.2 dated Dec. 12, 2013.

* cited by examiner

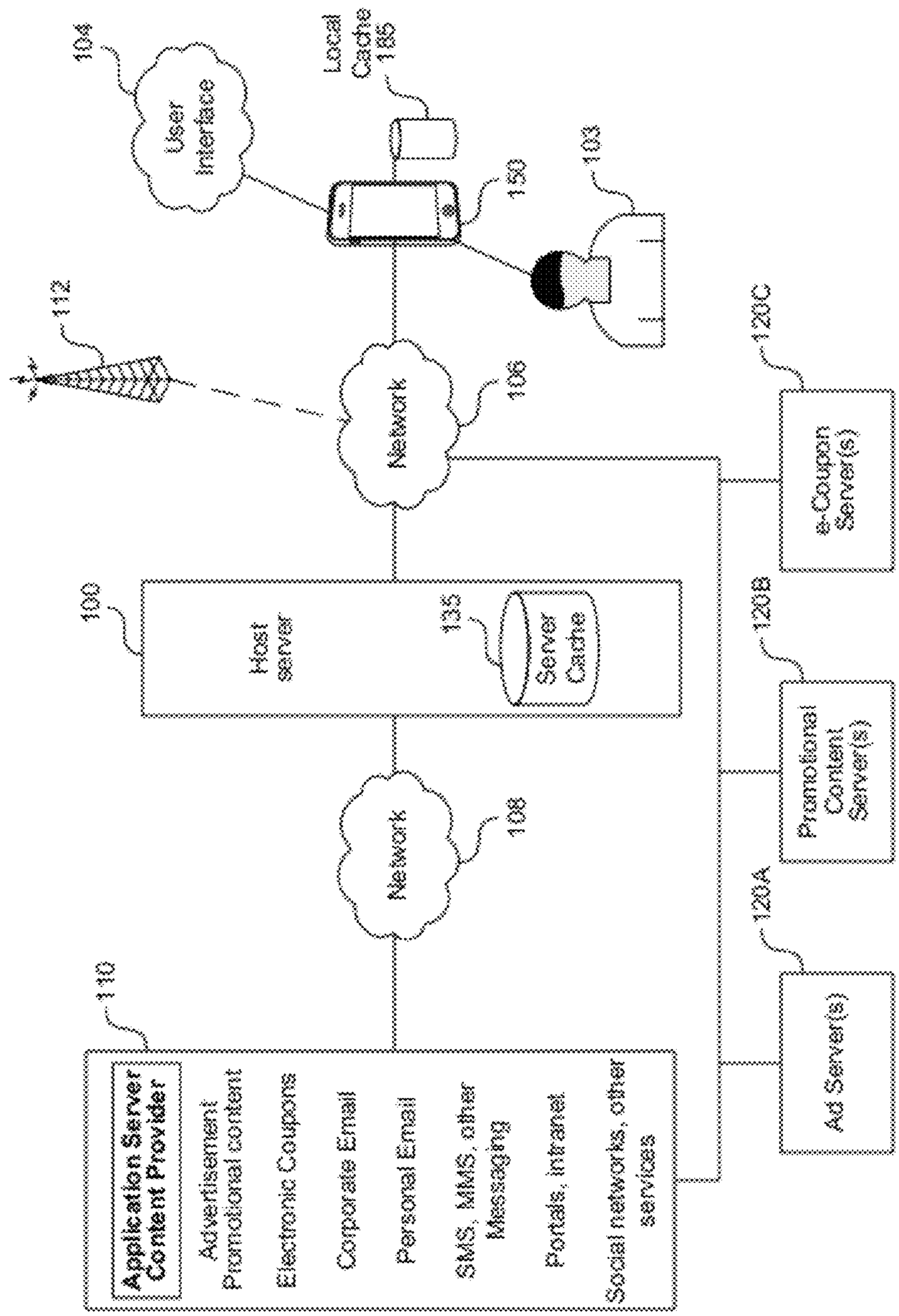

104
User Interface
Local Cache 185
150
103
112
Network
106
120C
e-Coupon Server(s)
100
Host server
135
Server Cache
120B
Promotional Content Server(s)
Network
108
120A
110
Ad Server(s)
Application Server Content Provider
Advertisement Promotional content
Electronic Coupons
Corporate Email
Personal Email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

FIG. 1A

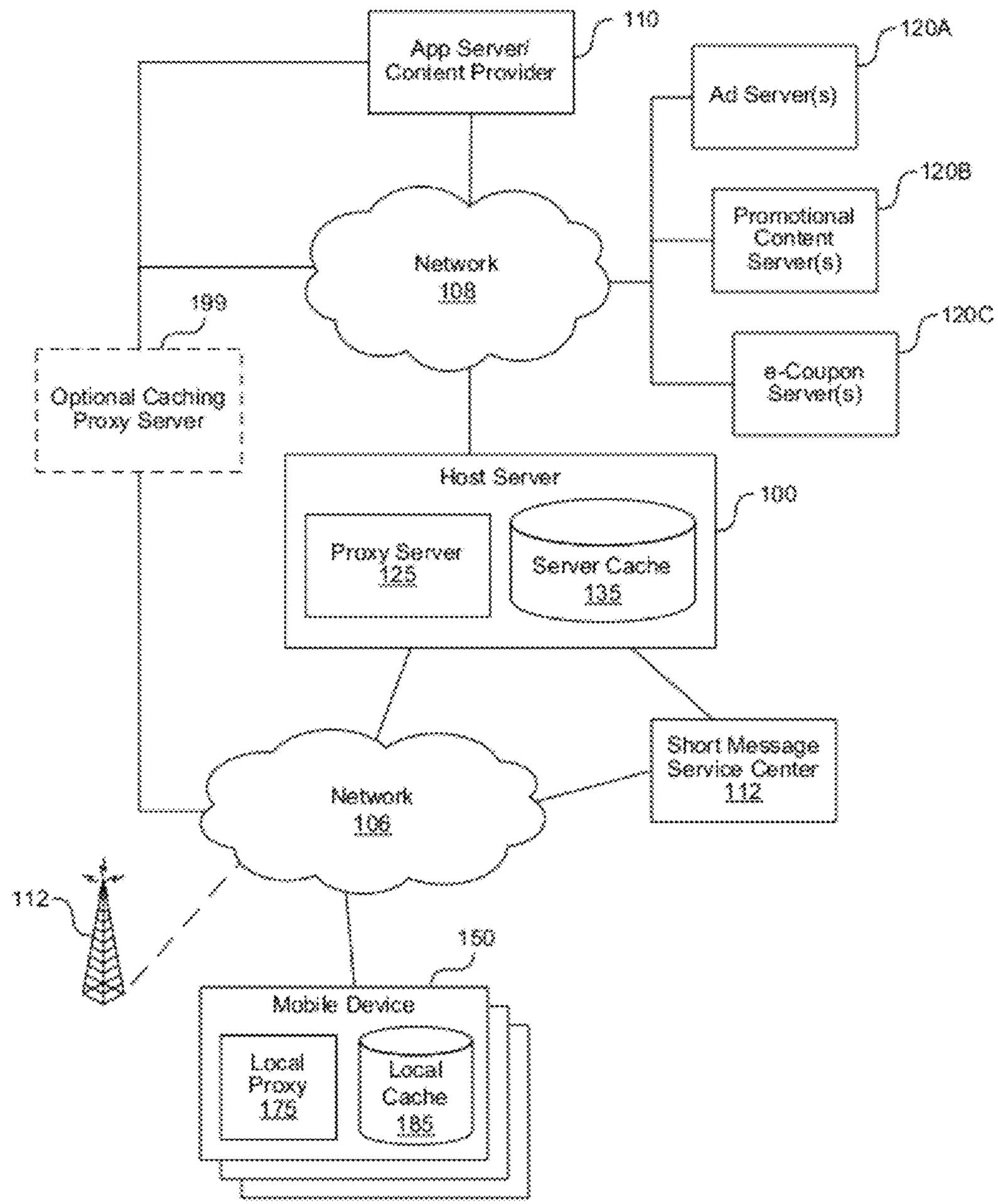

App Server/
Content Provider
110
120A
Ad Server(s)
120B
Promotional
Content
Server(s)
Network
108
199
120C
e-Coupon
Server(s)
Optional Caching
Proxy Server
Host Server
100
Proxy Server
125
Server Cache
135
Short Message
Service Center
112
Network
106
112
150
Mobile Device
Local
Proxy
175
Local
Cache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

FIG. 1B

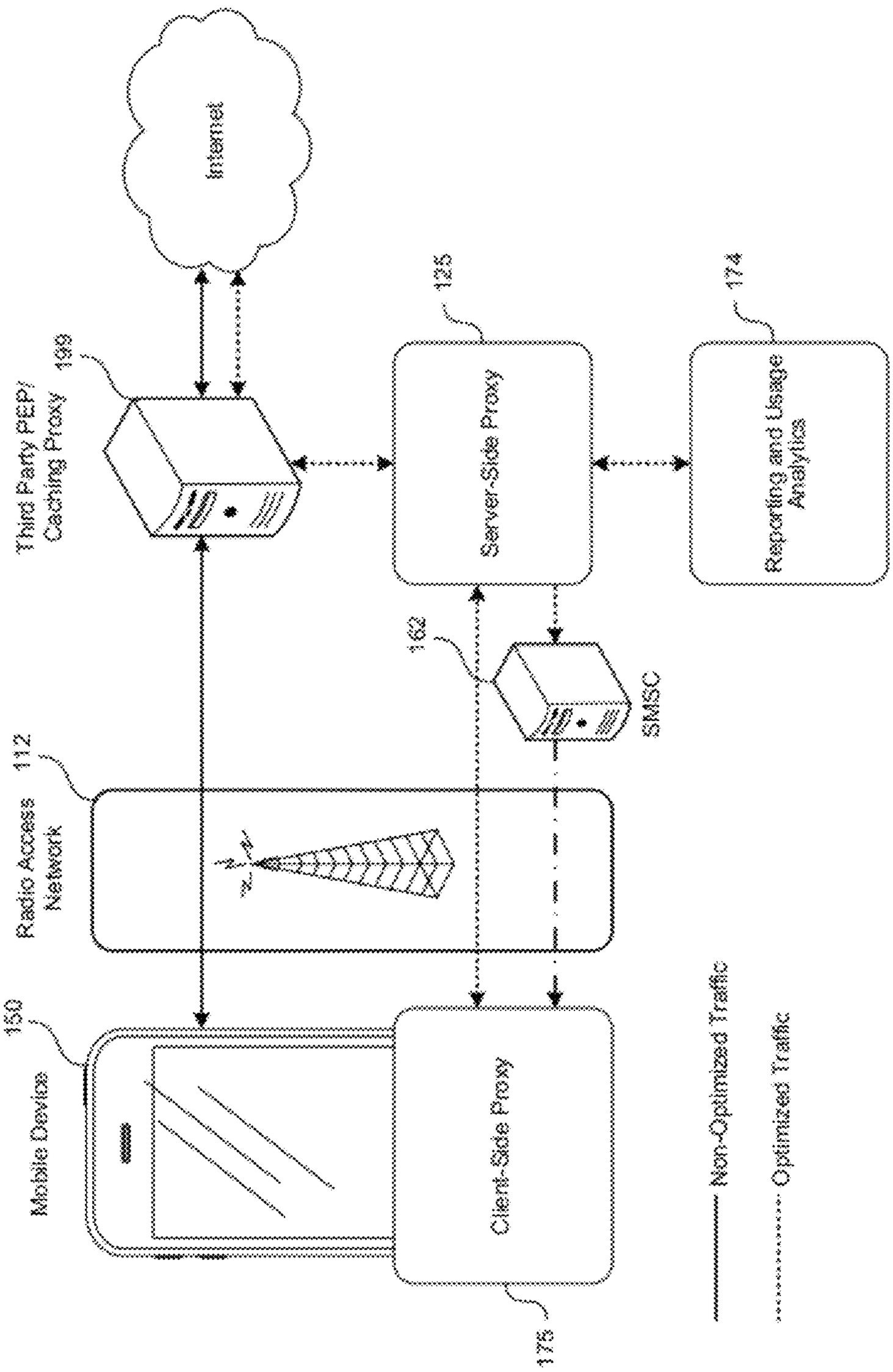

Internet
Third Party PEP/ Caching Proxy
199
Server-Side Proxy
125
Reporting and Usage Analytics
174
SMSC
162
Radio Access Network
112
Mobile Device
150
Client-Side Proxy
175
Non-Optimized Traffic
Optimized Traffic

FIG. 1C

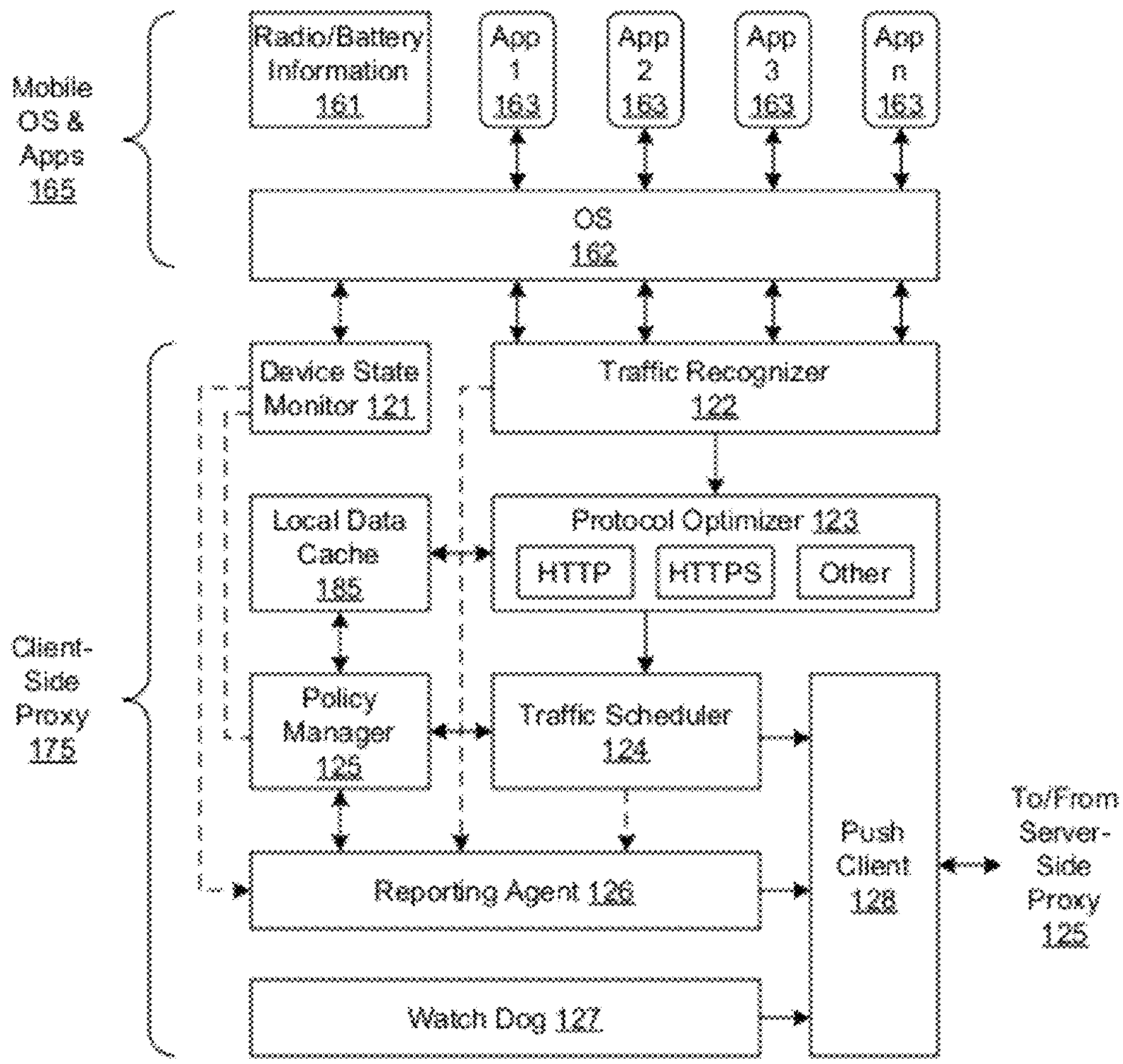

Mobile OS & Apps 165
Radio/Battery Information 161
App 1 163
App 2 163
App 3 163
App n 163
OS 162
Client-Side Proxy 175
Device State Monitor 121
Traffic Recognizer 122
Local Data Cache 185
Protocol Optimizer 123
HTTP
HTTPS
Other
Policy Manager 125
Traffic Scheduler 124
Reporting Agent 126
Watch Dog 127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

FIG. 1D

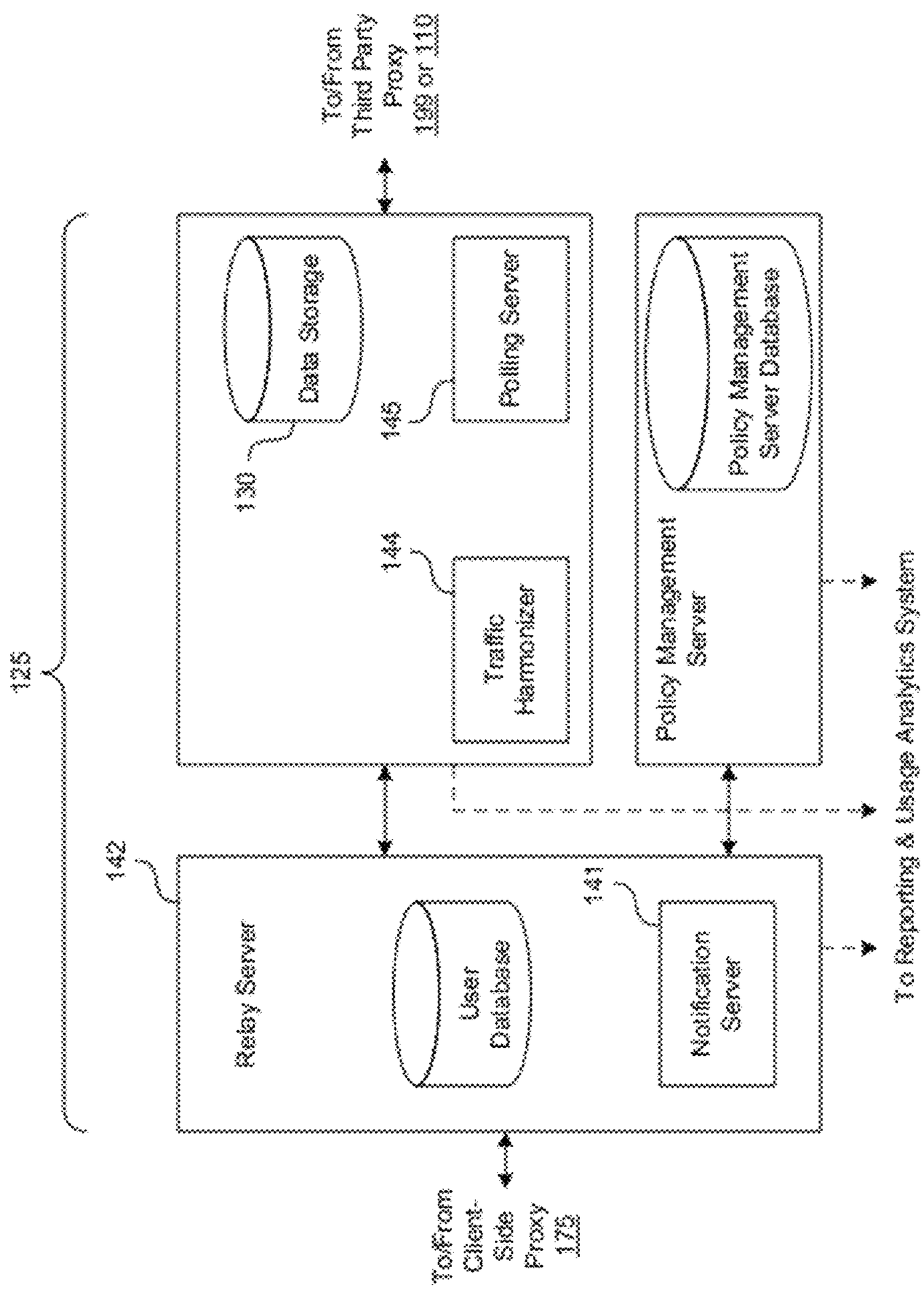

To/From Third Party Proxy 109 or 110
125
Data Storage
130
Polling Server
145
Traffic Harmonizer
144
Policy Management Server Database
Policy Management Server
142
Relay Server
User Database
141
Notification Server
To Reporting & Usage Analytics System
To/From Client-Side Proxy 175

FIG. 1E

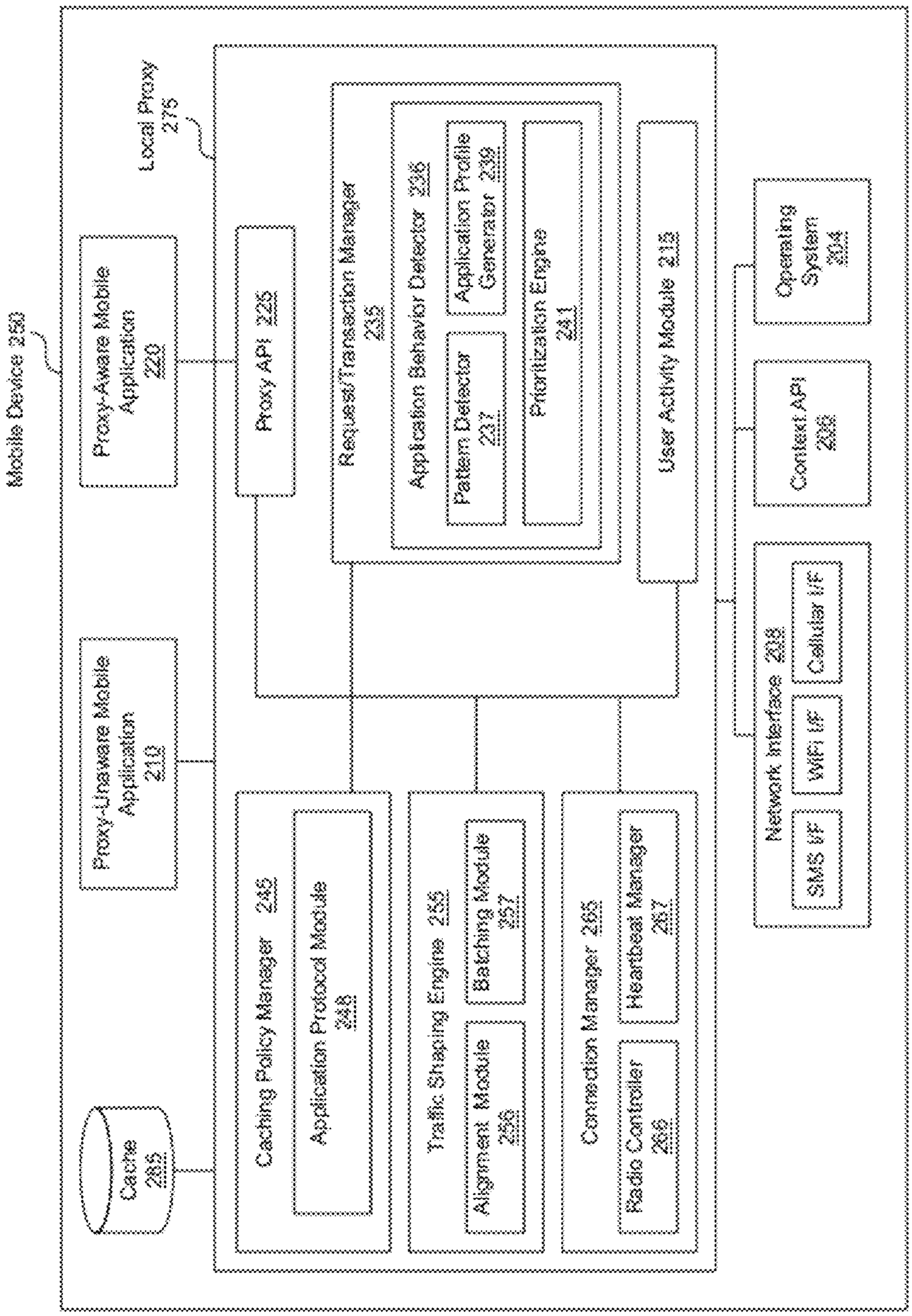

Mobile Device 250
Local Proxy 275
Proxy-Aware Mobile Application 220
Proxy-Unaware Mobile Application 210
Proxy API 225
Request/Transaction Manager 235
Application Behavior Detector 236
Pattern Detector 237
Application Profile Generator 239
Prioritization Engine 241
User Activity Module 215
Operating System 204
Context API 205
Network Interface 208
SMS I/F
WiFi I/F
Cellular I/F
Cache 285
Caching Policy Manager 245
Application Protocol Module 248
Traffic Shaping Engine 255
Alignment Module 258
Batching Module 257
Connection Manager 265
Radio Controller 266
Heartbeat Manager 267

FIG. 2A

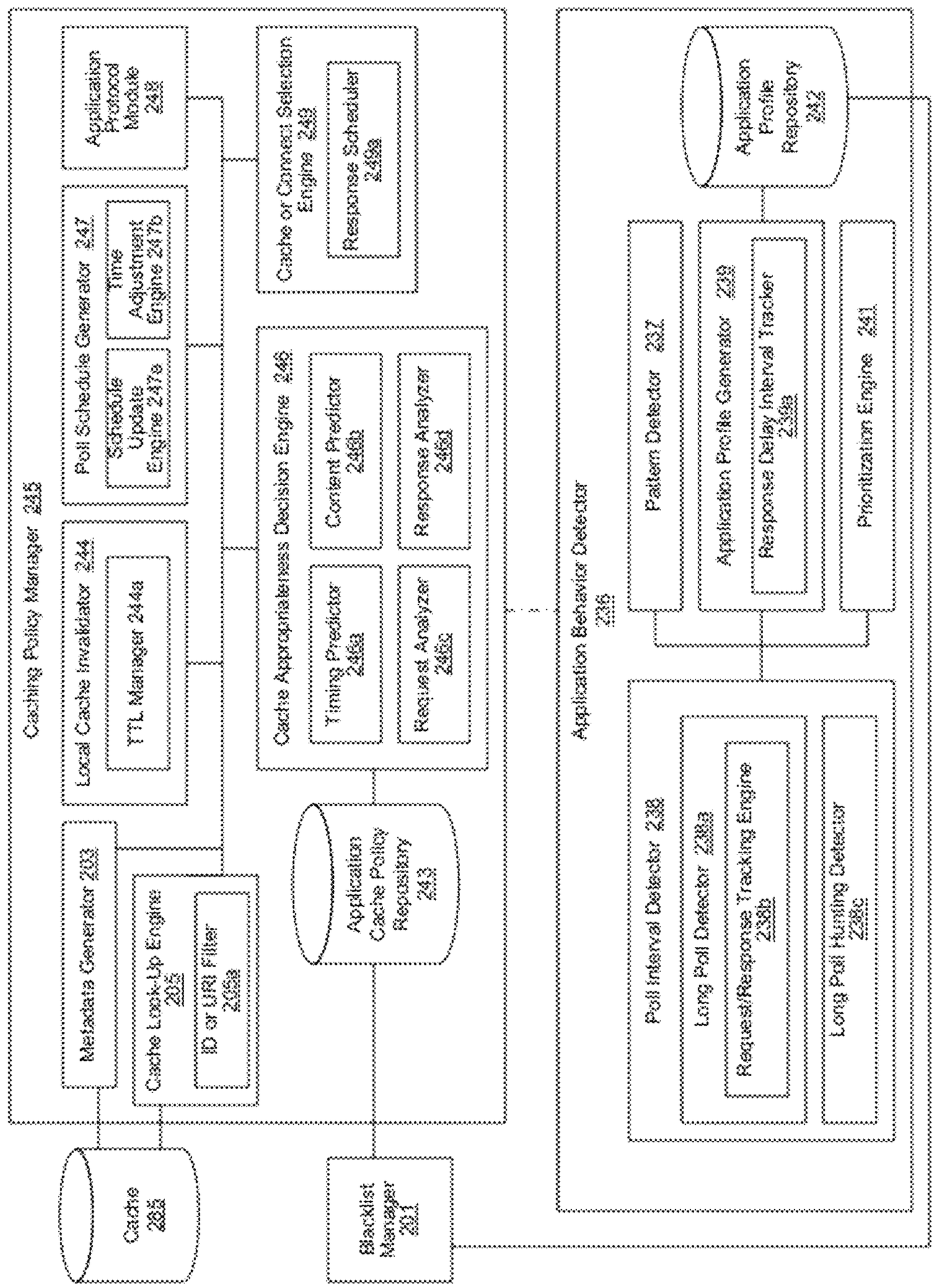

Caching Policy Manager 245
Application Protocol Module 248
Poll Schedule Generator 247
Schedule Update Engine 247a
Time Adjustment Engine 247b
Cache or Connect Selection Engine 249
Response Scheduler 249a
Local Cache Invalidator 244
TTL Manager 244a
Cache Appropriateness Decision Engine 246
Content Predictor 246b
Timing Predictor 246a
Response Analyzer 246d
Request Analyzer 246c
Metadata Generator 203
Cache Look-Up Engine 205
ID or URI Filter 205a
Application Cache Policy Repository 243
Cache 255
Blacklist Manager 251
Application Behavior Detector 236
Application Profile Repository 242
Pattern Detector 237
Application Profile Generator 239
Response Delay Interval Tracker 239a
Prioritization Engine 241
Poll Interval Detector 238
Long Poll Detector 238a
Request/Response Tracking Engine 238b
Long Poll Hunting Detector 238c

FIG. 2B

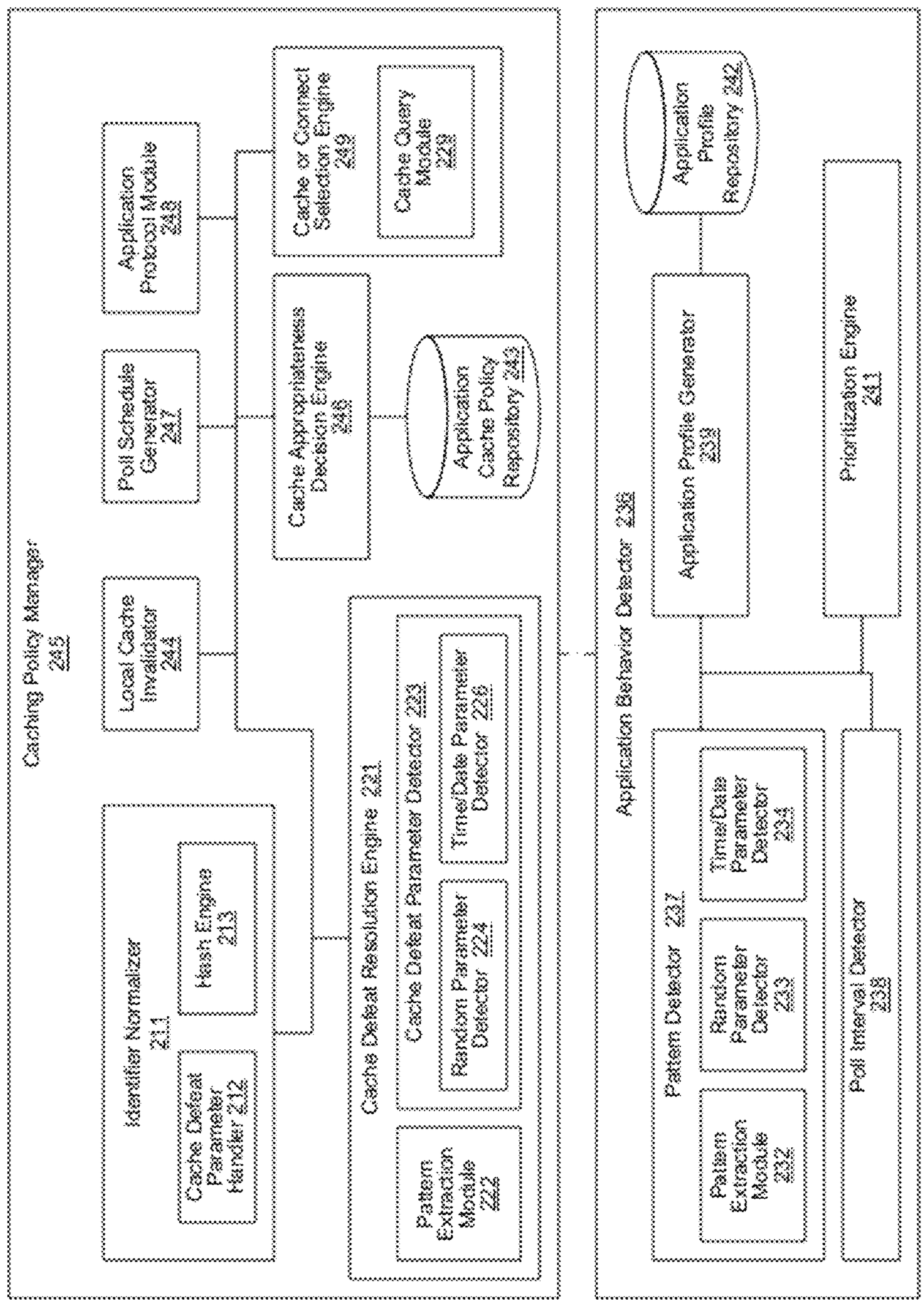

Caching Policy Manager 246
Identifier Normalizer 211
Cache Defeat Parameter Handler 212
Hash Engine 213
Local Cache Invalidator 244
Poll Schedule Generator 247
Application Protocol Module 248
Cache Appropriateness Decision Engine 246
Cache or Connect Selection Engine 249
Cache Query Module 229
Application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

FIG. 2C

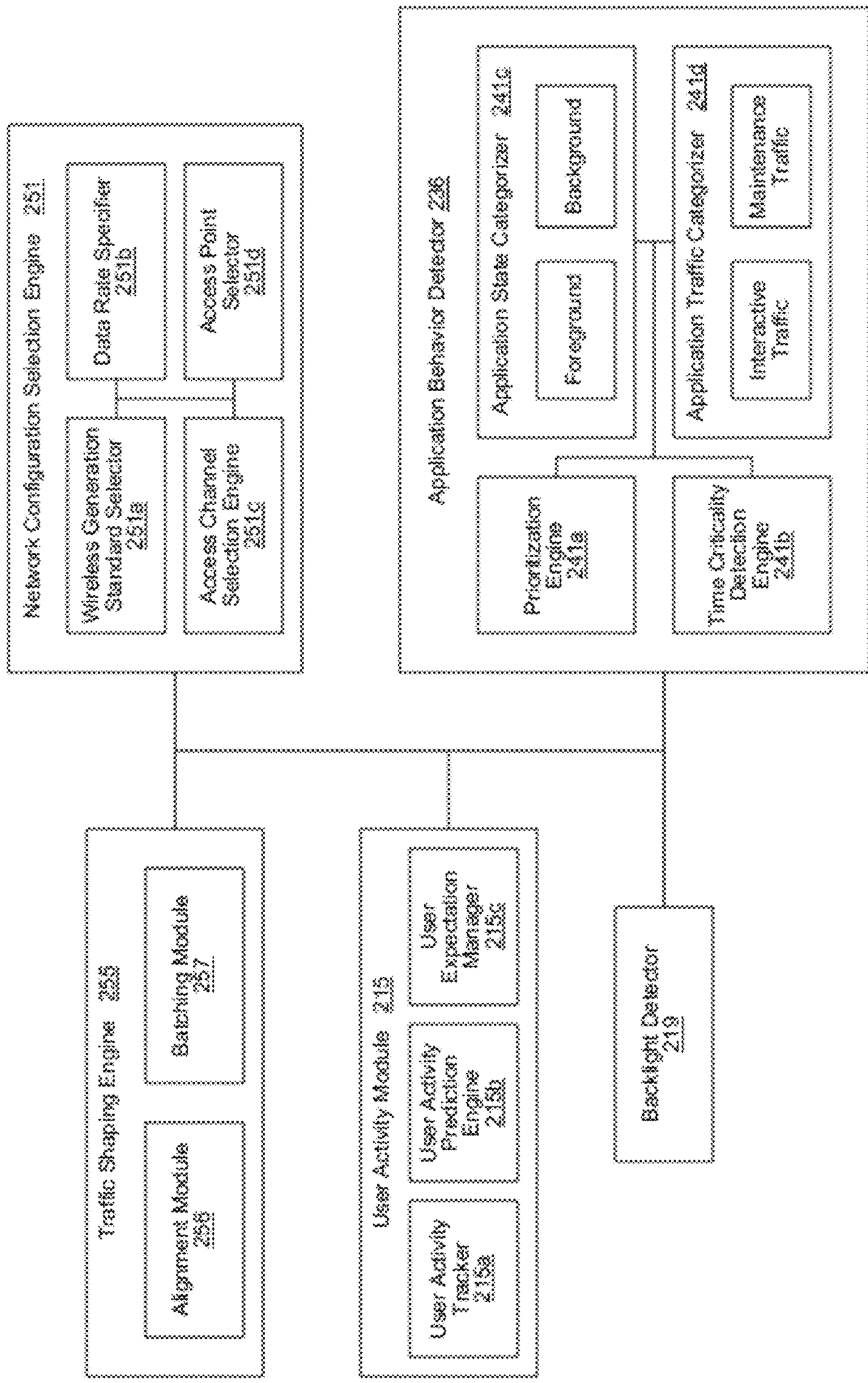

Network Configuration Selection Engine 251
Wireless Generation Standard Selector 251a
Data Rate Specifier 251b
Access Channel Selection Engine 251c
Access Point Selector 251d
Application Behavior Detector 236
Application State Categorizer 241c
Foreground
Background
Application Traffic Categorizer 241d
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. 2D

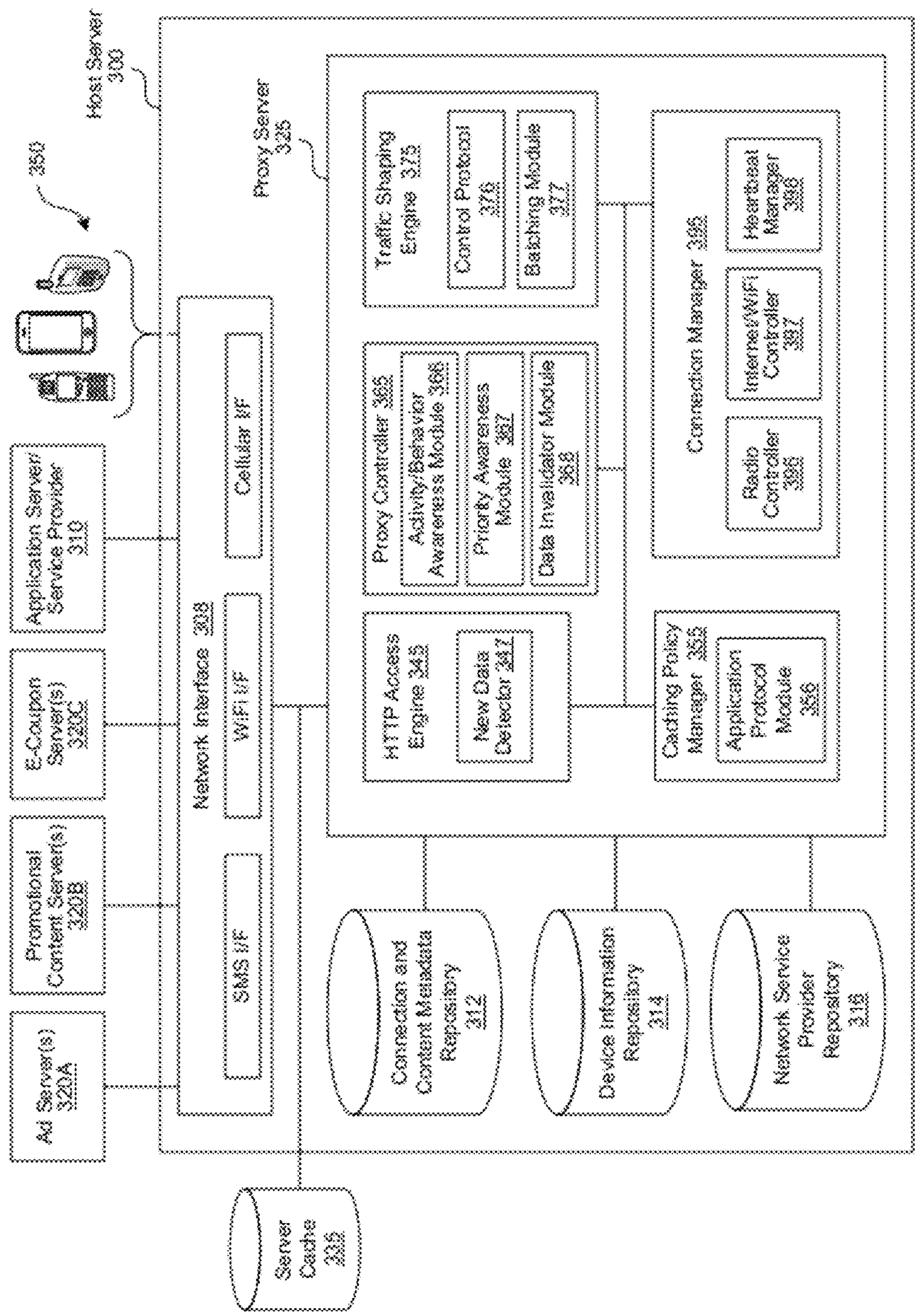

Host Server 300
350
Proxy Server 325
Traffic Shaping Engine 375
Control Protocol 376
Batching Module 377
Proxy Controller 365
Activity/Behavior Awareness Module 366
Priority Awareness Module 367
Data Invalidator Module 368
HTTP Access Engine 345
New Data Detector 347
Connection Manager 355
Heartbeat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

FIG. 3A

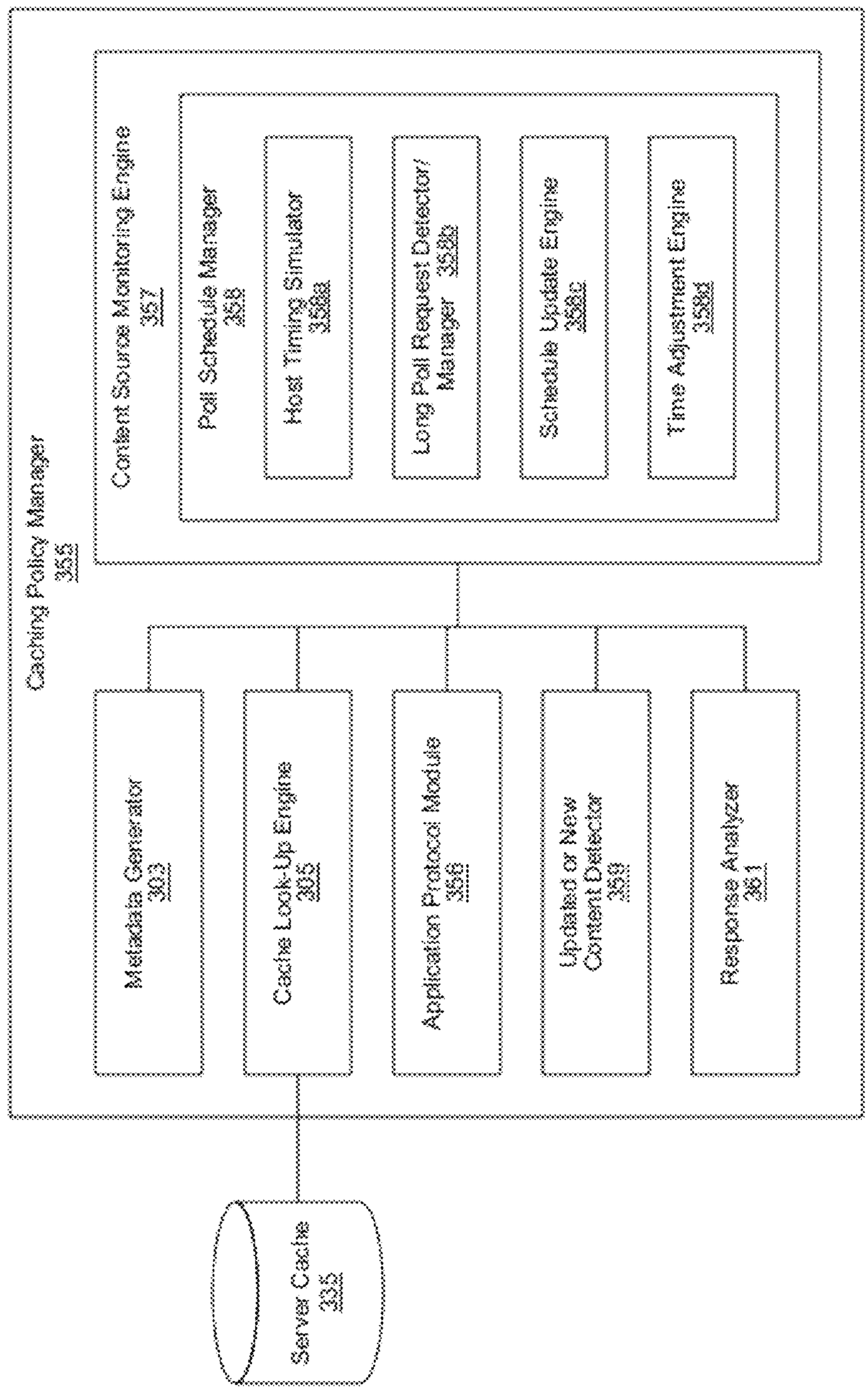

Caching Policy Manager
356
Content Source Monitoring Engine
357
Poll Schedule Manager
358
Host Timing Simulator
358a
Long Poll Request Detector/
Manager 358b
Schedule Update Engine
358c
Time Adjustment Engine
358d
Metadata Generator
303
Cache Look-Up Engine
305
Application Protocol Module
356
Updated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

FIG. 3B

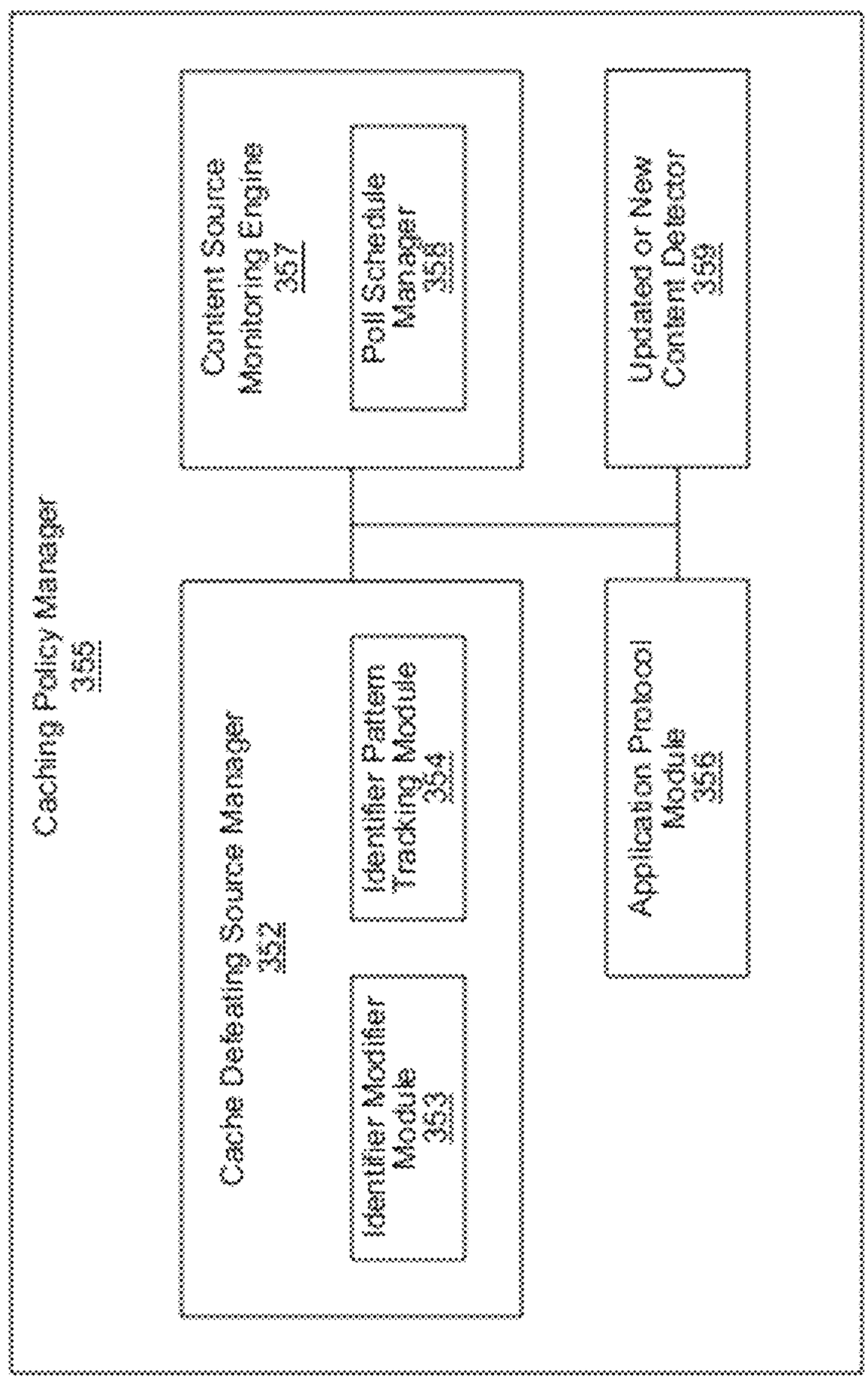

Caching Policy Manager
365
Content Source
Monitoring Engine
357
Poll Schedule
Manager
358
Updated or New
Content Detector
359
Cache Deleating Source Manager
352
Identifier Pattern
Tracking Module
354
Identifier Modifier
Module
353
Application Protocol
Module
356

FIG. 3C

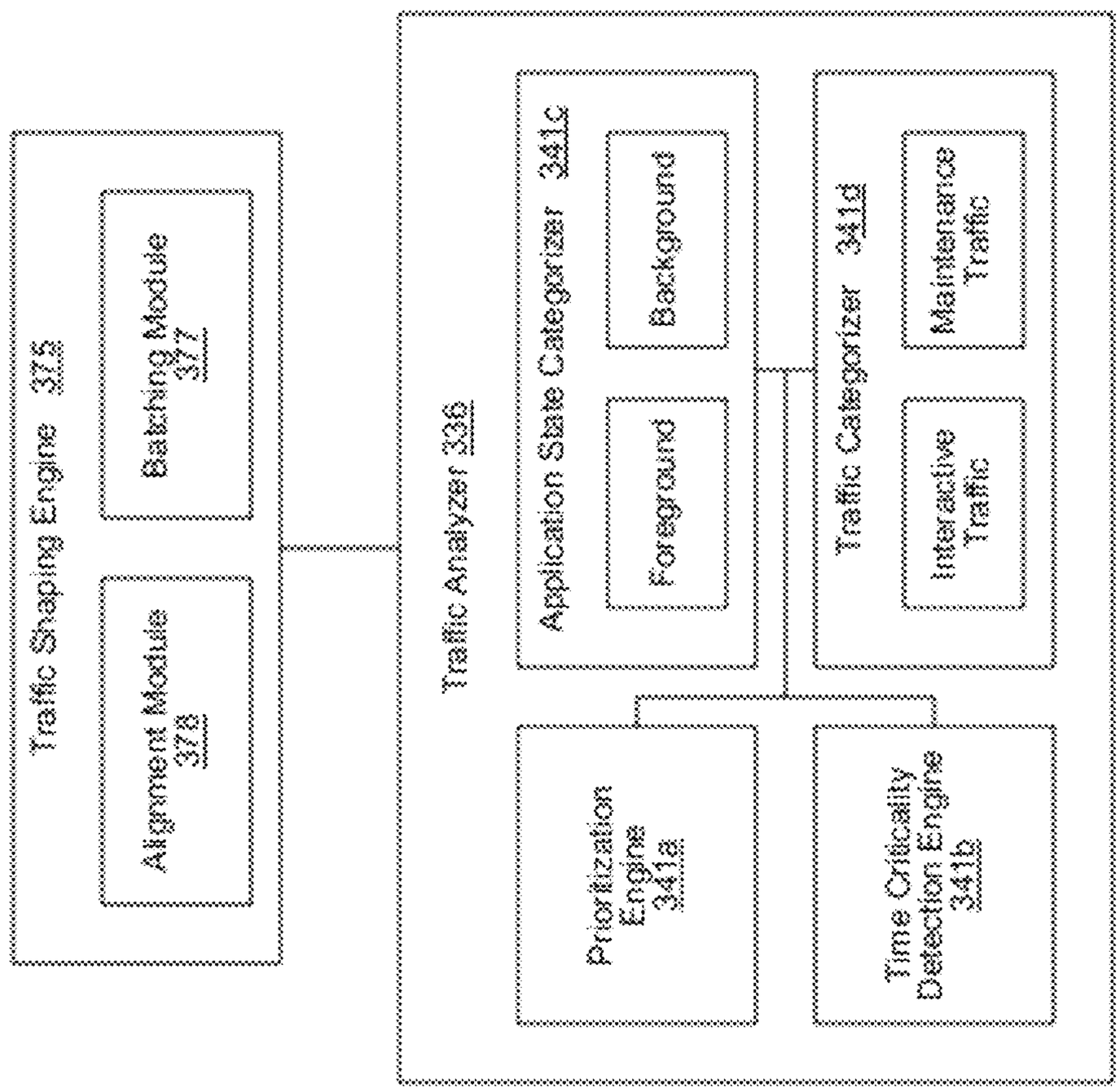

Traffic Shaping Engine 375
Alignment Module 378
Batching Module 377
Traffic Analyzer 336
Application State Categorizer 341c
Foreground
Background
Traffic Categorizer 341d
Interactive Traffic
Maintenance Traffic
Prioritization Engine 341a
Time Criticality Detection Engine 341b

FIG. 3D

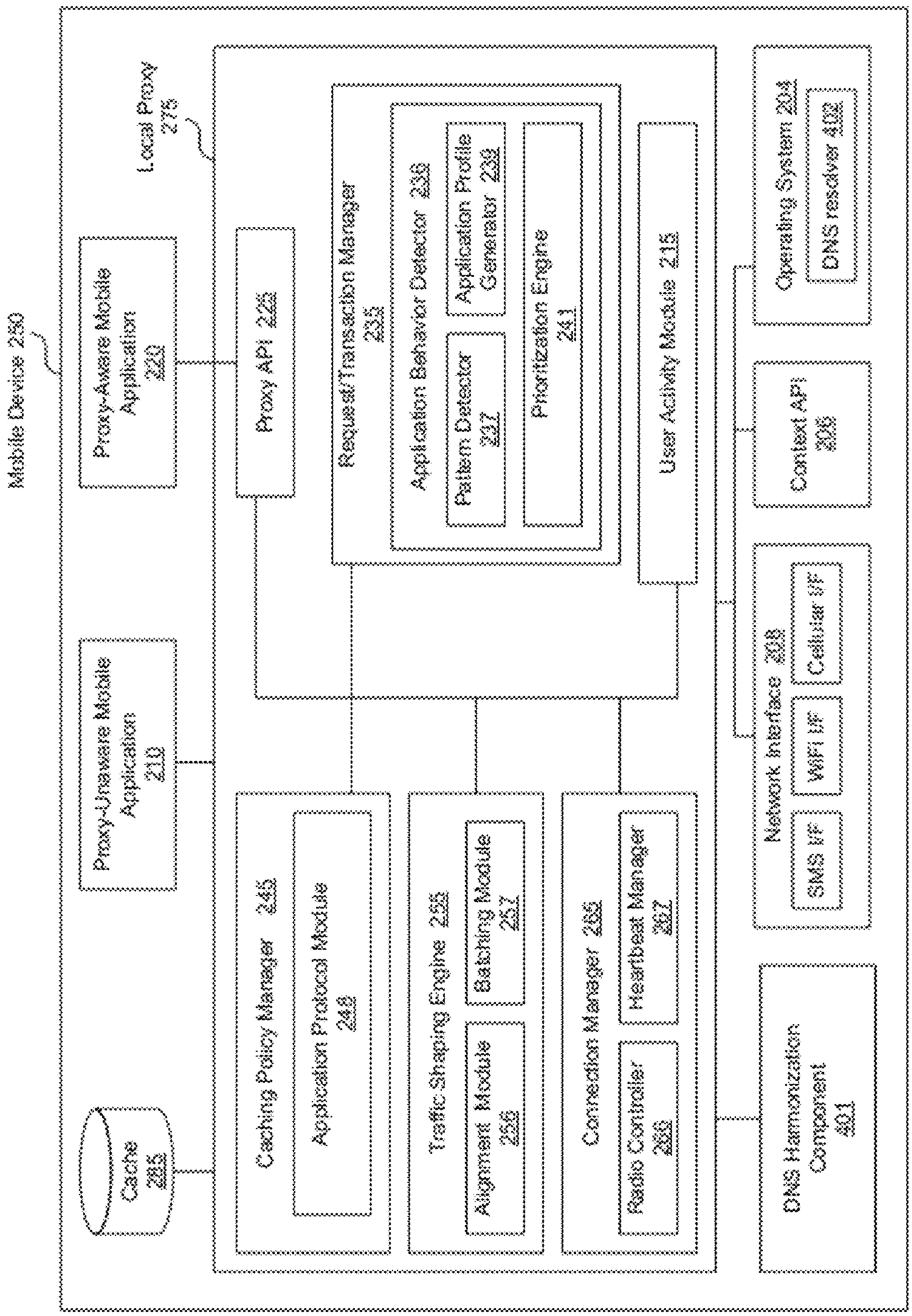

Mobile Device 250
Local Proxy 275
Proxy-Aware Mobile Application 220
Proxy-Unaware Mobile Application 210
Proxy API 225
Request/Transaction Manager 235
Application Behavior Detector 236
Pattern Detector 237
Application Profile Generator 239
Prioritization Engine 241
User Activity Module 215
Operating System 204
DNS resolver 402
Context API 205
Network Interface 208
SMS I/F
WiFi I/F
Cellular I/F
Cache 285
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
DNS Harmonization Component 401

FIG. 4

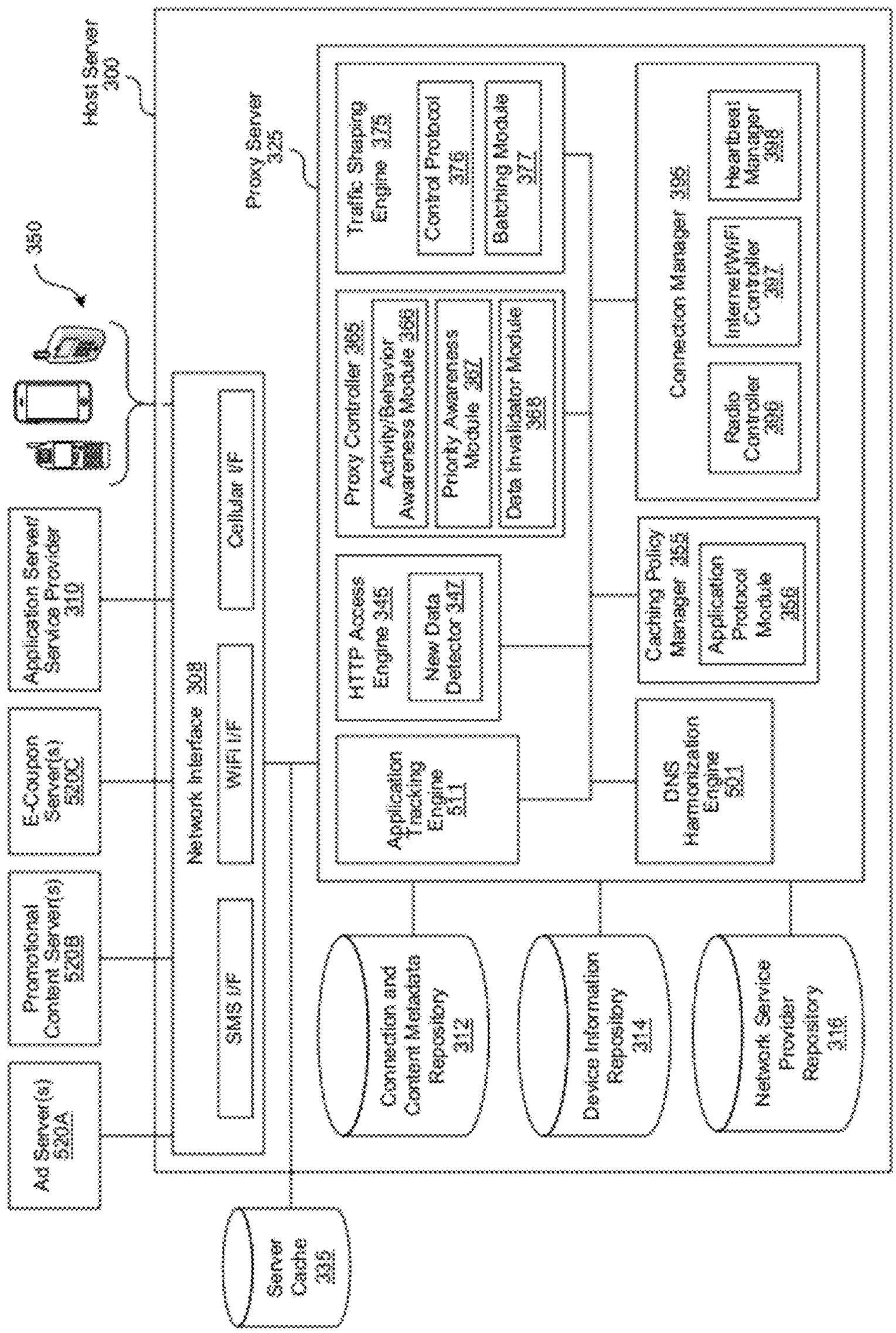

Host Server 300
360
Application Server/ Service Provider 310
E-Coupon Server(s) 320C
Promotional Content Server(s) 320B
Ad Server(s) 320A
Network Interface 308
Cellular I/F
WiFi I/F
SMS I/F
Proxy Server 325
Traffic Shaping Engine 375
Control Protocol 376
Batching Module 377
Proxy Controller 365
Activity/Behavior Awareness Module 366
Priority Awareness Module 367
Data Invalidator Module 368
HTTP Access Engine 345
New Data Detector 347
Application Tracking Engine 511
Connection Manager 395
Heartbeat Manager 398
Internet/WiFi Controller 397
Radio Controller 396
Caching Policy Manager 355
Application Protocol Module 356
DNS Harmonization Engine 501
Connection and Content Metadata Repository 312
Device Information Repository 314
Network Service Provider Repository 316
Server Cache 305

FIG. 5A

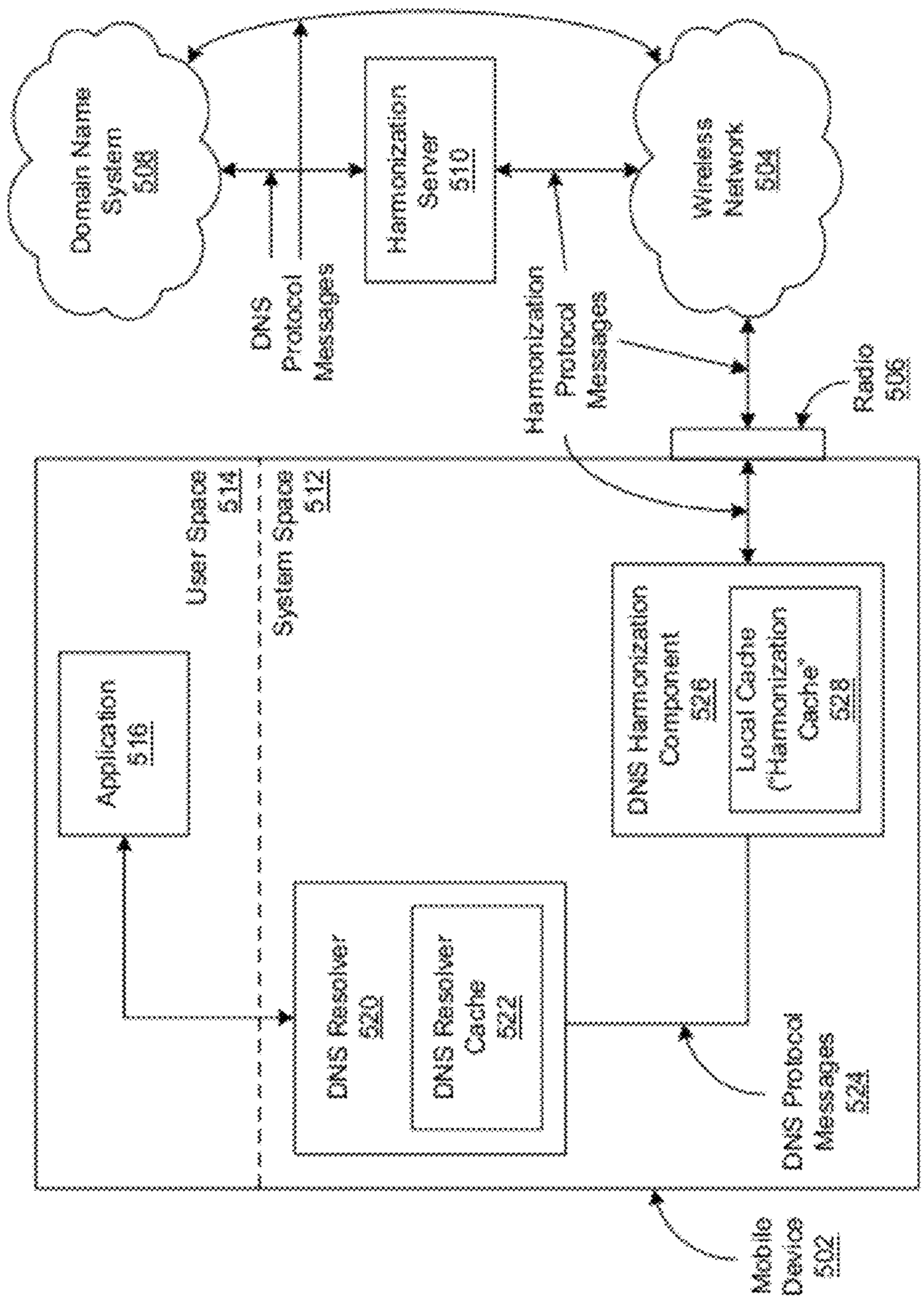

Domain Name System 508
Harmonization Server 510
Wireless Network 504
DNS Protocol Messages
Harmonization Protocol Messages
Radio 506
User Space 514
System Space 512
Application 516
DNS Harmonization Component 526
Local Cache ("Harmonization Cache" 528
DNS Resolver 520
DNS Resolver Cache 522
DNS Protocol Messages 524
Mobile Device 502

FIG. 5B

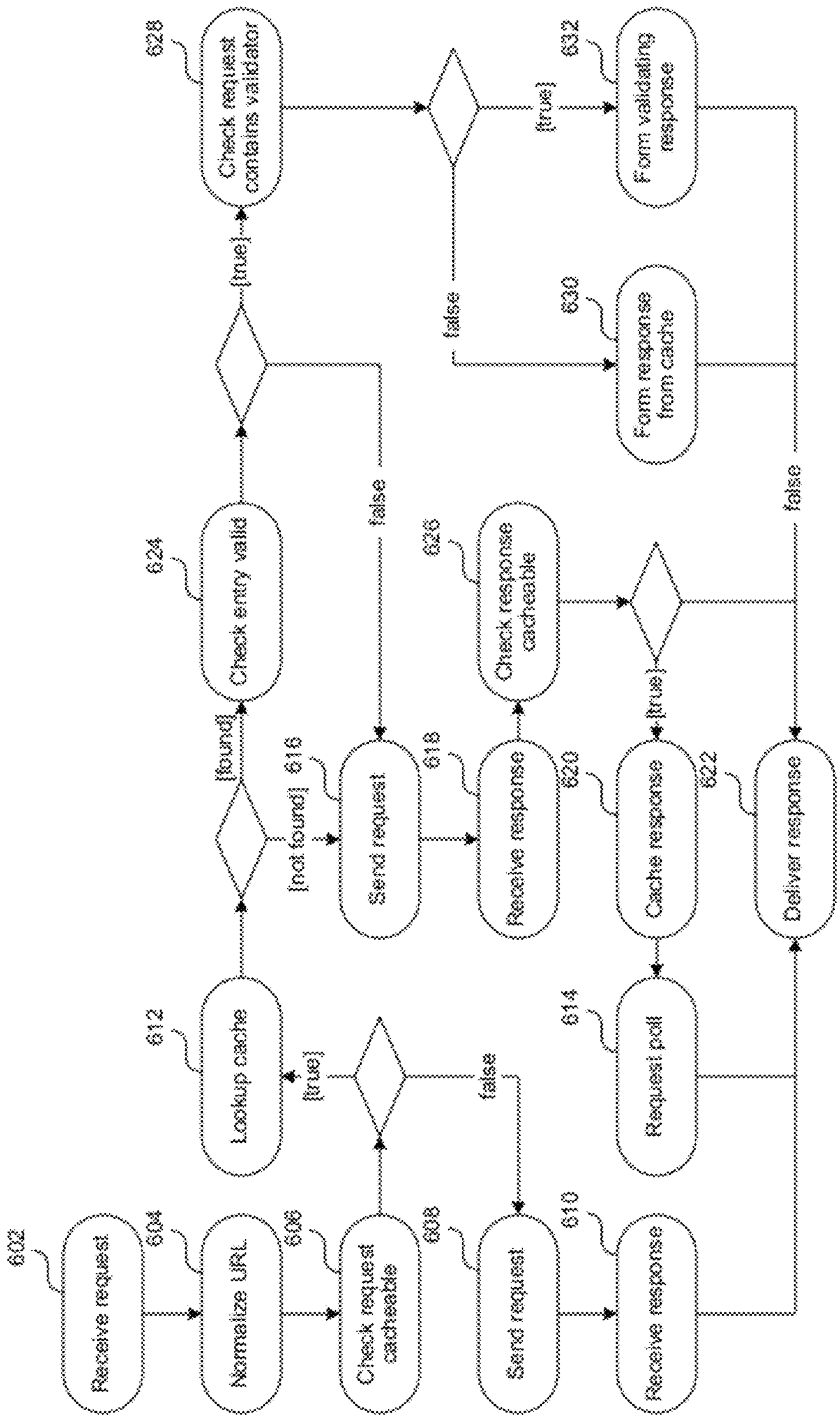

602
Receive request
604
Normalize URL
606
Check request cacheable
608
Send request
610
Receive response
612
Lookup cache
[true]
false
614
Request pull
[found]
[not found] 616
Send request
618
Receive response
620
Cache response
622
Deliver response
624
Check entry valid
[true]
false
626
Check response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

FIG. 6A

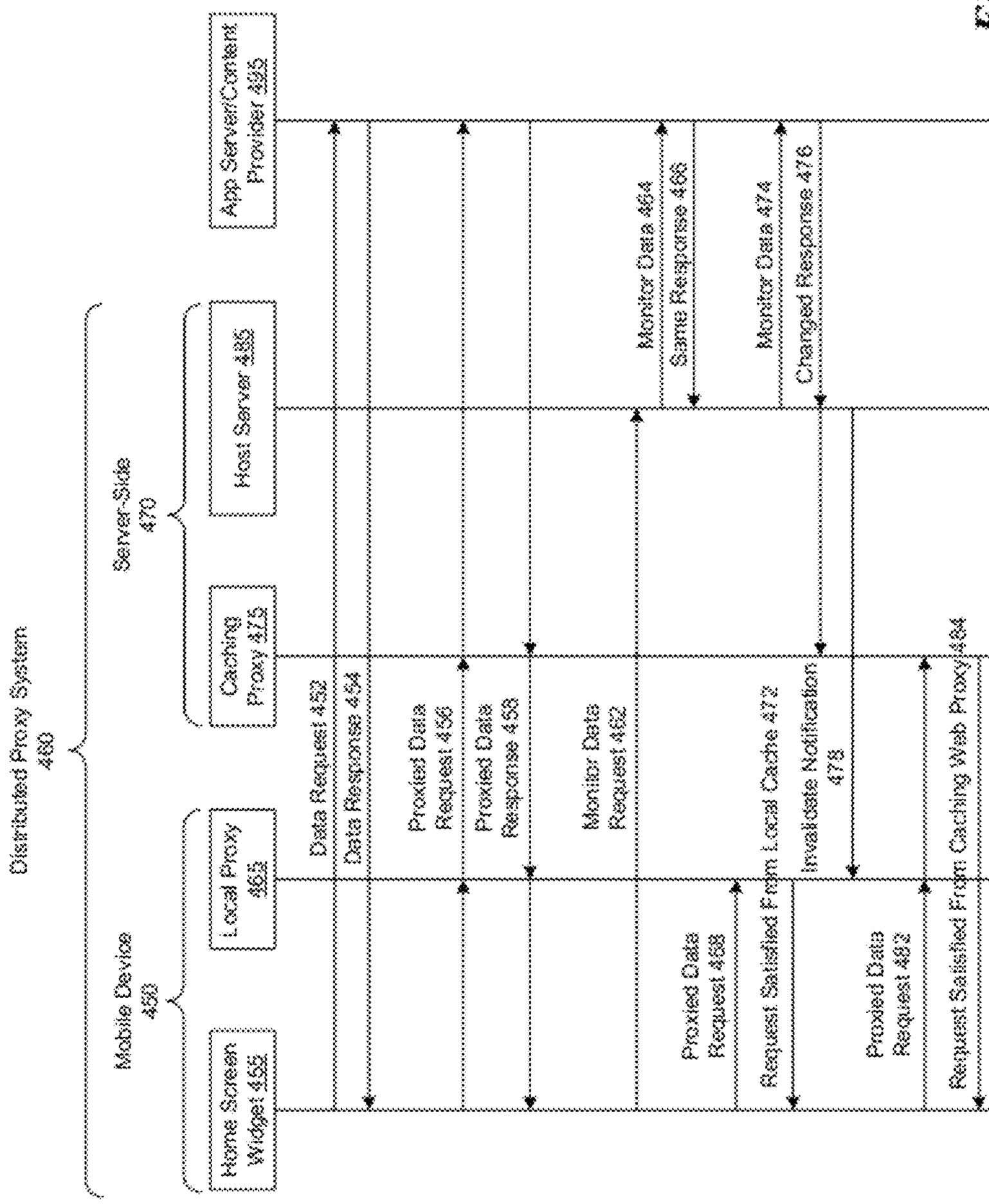

Distributed Proxy System
480
Mobile Device
450
Server-Side
470
Home Screen
Widget 455
Local Proxy
465
Caching
Proxy 475
Host Server 485
App Server/Content
Provider 495
Data Request 452
Data Response 454
Proxied Data
Request 456
Proxied Data
Response 458
Monitor Data
Request 462
Monitor Data 464
Same Response 466
Monitor Data 474
Changed Response 476
Proxied Data
Request 468
Request Satisfied From Local Cache 472
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, polls 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	Caches 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, polls 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

1100

Time	Local Proxy 1165	Server 1185	Action
0m0s	Request #1 to example.com		DNS entry retrieved and cached by Harmonization component
7m0s	Request #2 to example.com		DNS entry taken from Resolver cache
10m0s			Resolver cache entry for example.com expires
14m0s	Request #3 to example.com		DNS entry taken from Harmonization cache
14m5s	Send polling task to server	Start polling example.com	Harmonization cache entry refcount++ (becomes 1)
21m0s	Request #4 to example.com		DNS entry taken from Harmonization cache; response taken from Harmonization cache
21m5s		Continue polling example.com	
...			
60m0s			This is when the Harmonization cache entry would expire if there were no polled resources tied to it
...			
70m5s		Receive new content in poll response, send cache invalidate to Harmonization component at client	
70m10s	Cache invalidate received		Harmonization cache entry refcount-- (becomes 0); example.com removed from Harmonization cache
77m0s	Request #12 to example.com		DNS entry retrieved and cached by Harmonization component at client

FIG. 11

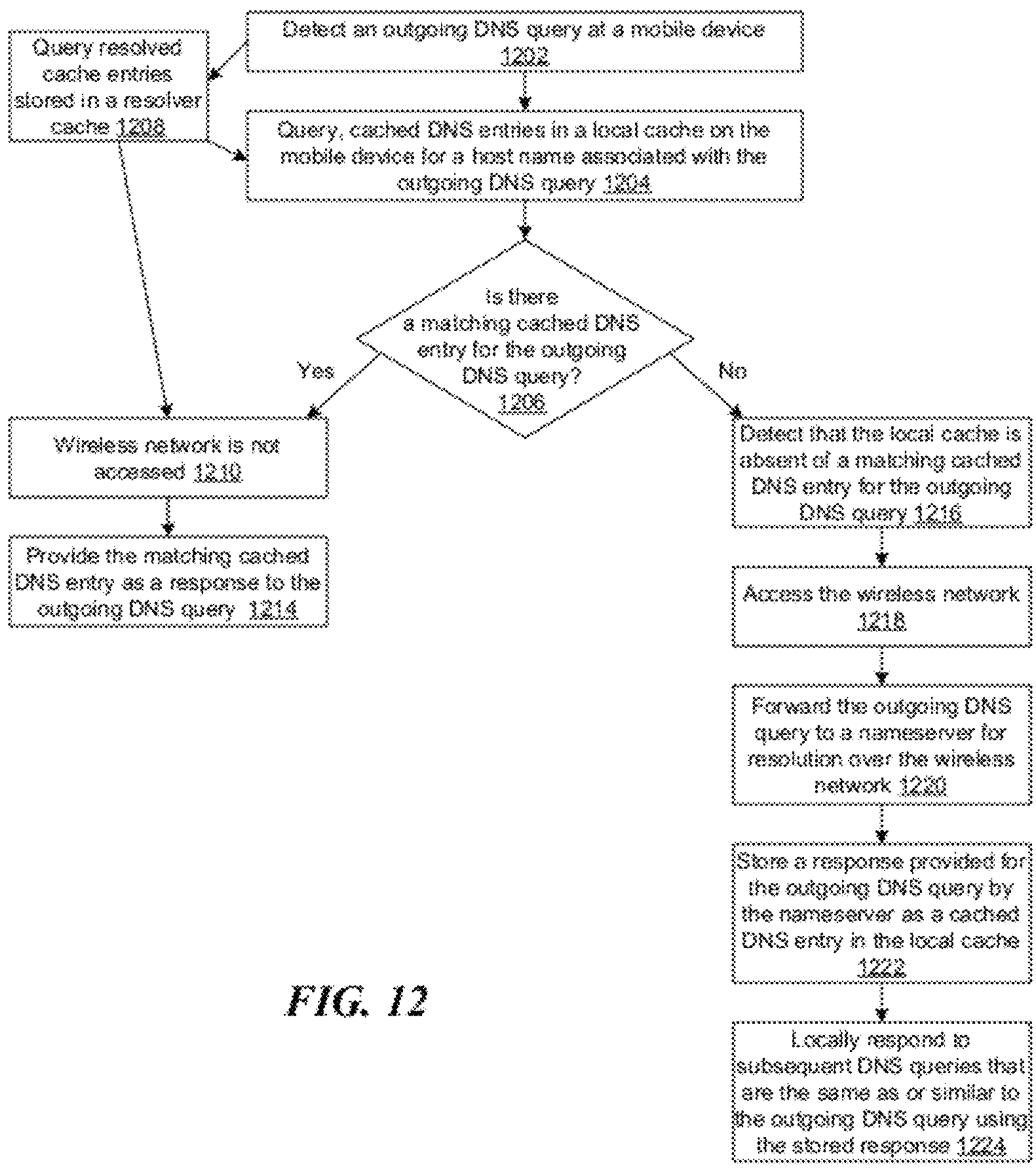

Detect an outgoing DNS query at a mobile device 1202
Query resolved cache entries stored in a resolver cache 1208
Query, cached DNS entries in a local cache on the mobile device for a host name associated with the outgoing DNS query 1204
Is there a matching cached DNS entry for the outgoing DNS query? 1206
Yes
No
Wireless network is not accessed 1210
Provide the matching cached DNS entry as a response to the outgoing DNS query 1214
Detect that the local cache is absent of a matching cached DNS entry for the outgoing DNS query 1216
Access the wireless network 1218
Forward the outgoing DNS query to a nameserver for resolution over the wireless network 1220
Store a response provided for the outgoing DNS query by the nameserver as a cached DNS entry in the local cache 1222
Locally respond to subsequent DNS queries that are the same as or similar to the outgoing DNS query using the stored response 1224

FIG. 12

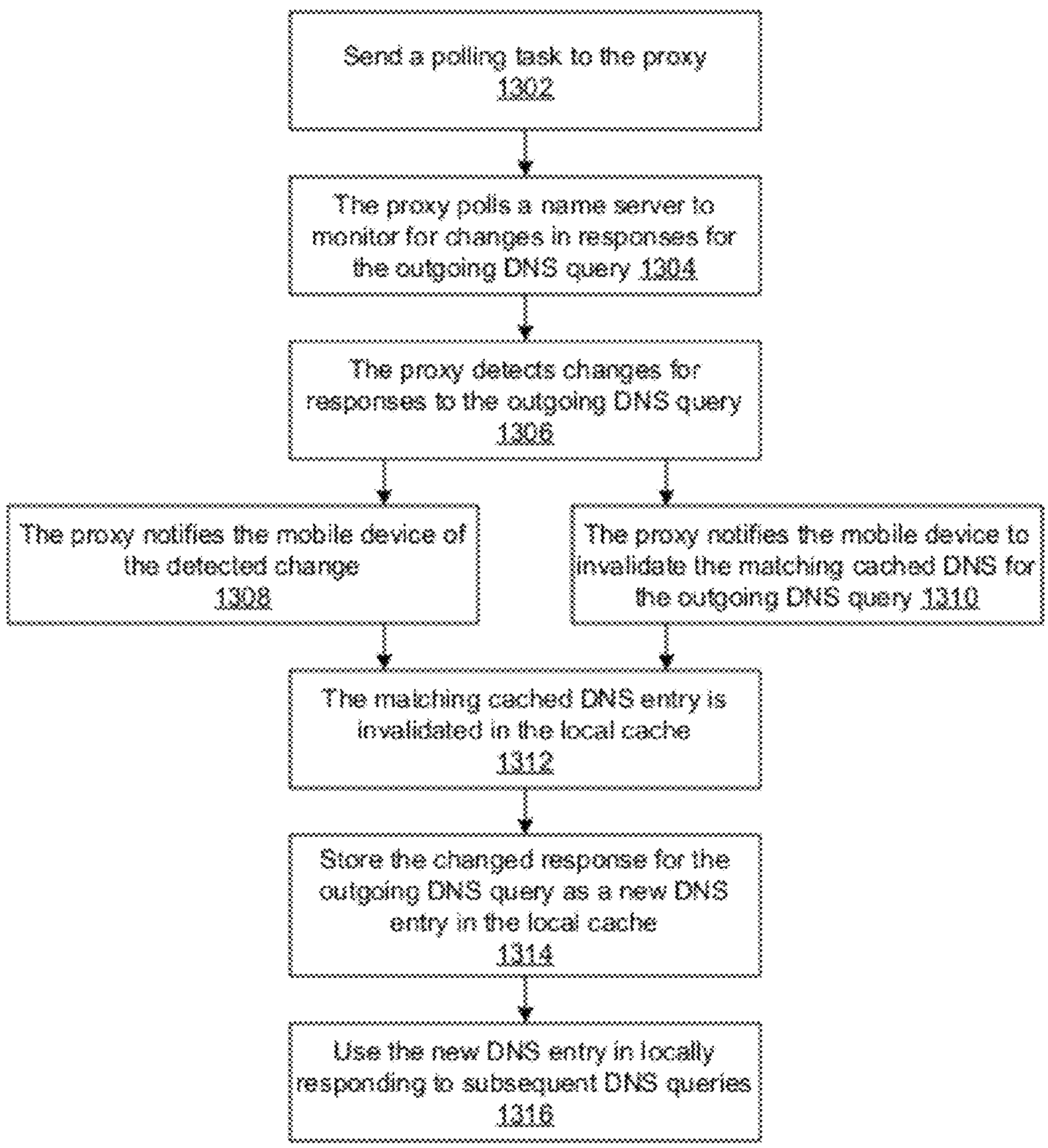

Send a polling task to the proxy 1302
The proxy polls a name server to monitor for changes in responses for the outgoing DNS query 1304
The proxy detects changes for responses to the outgoing DNS query 1306
The proxy notifies the mobile device of the detected change 1308
The proxy notifies the mobile device to invalidate the matching cached DNS for the outgoing DNS query 1310
The matching cached DNS entry is invalidated in the local cache 1312
Store the changed response for the outgoing DNS query as a new DNS entry in the local cache 1314
Use the new DNS entry in locally responding to subsequent DNS queries 1316

FIG. 13

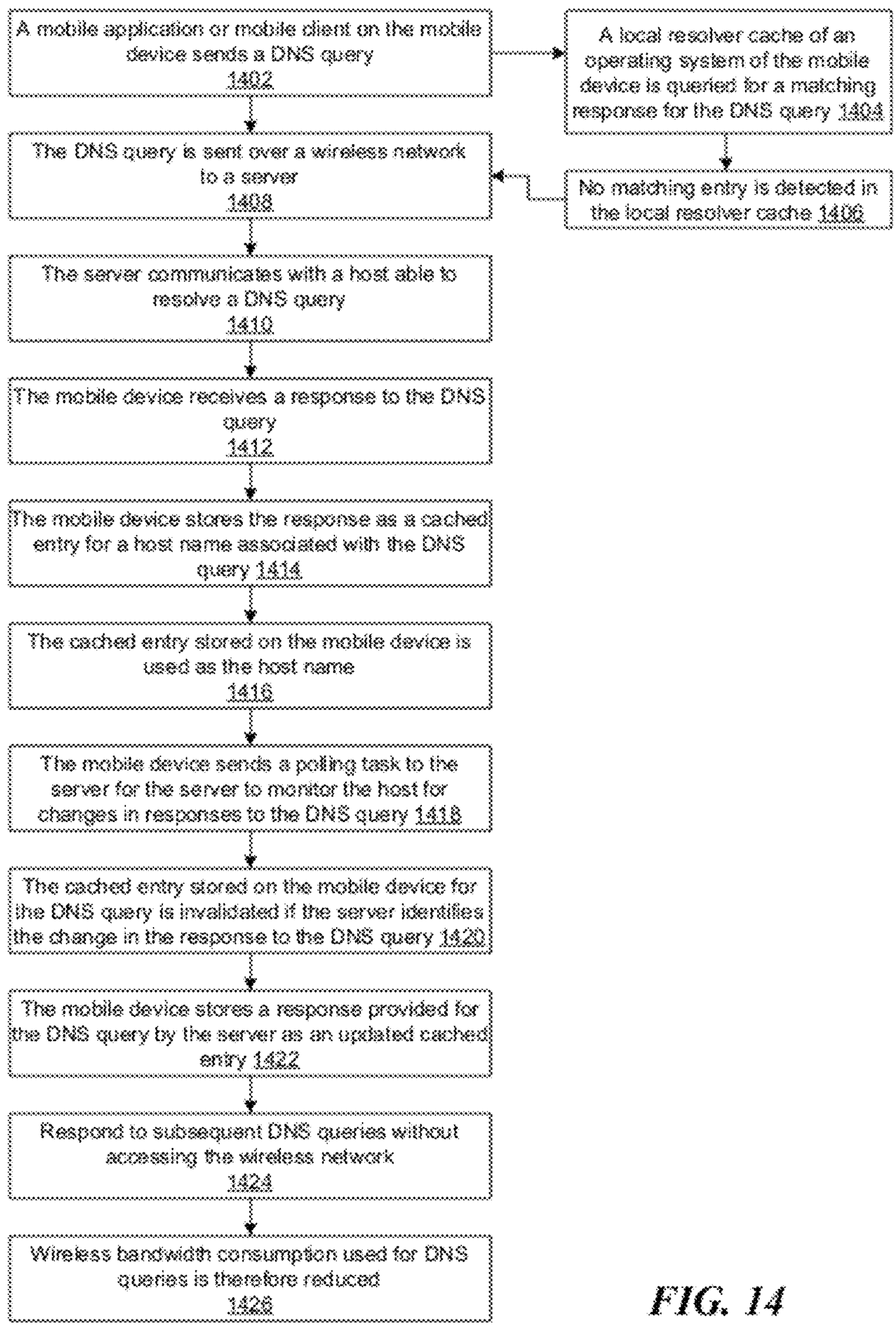

A mobile application or mobile client on the mobile device sends a DNS query 1402
A local resolver cache of an operating system of the mobile device is queried for a matching response for the DNS query 1404
No matching entry is detected in the local resolver cache 1406
The DNS query is sent over a wireless network to a server 1408
The server communicates with a host able to resolve a DNS query 1410
The mobile device receives a response to the DNS query 1412
The mobile device stores the response as a cached entry for a host name associated with the DNS query 1414
The cached entry stored on the mobile device is used as the host name 1416
The mobile device sends a polling task to the server for the server to monitor the host for changes in responses to the DNS query 1418
The cached entry stored on the mobile device for the DNS query is invalidated if the server identifies the change in the response to the DNS query 1420
The mobile device stores a response provided for the DNS query by the server as an updated cached entry 1422
Respond to subsequent DNS queries without accessing the wireless network 1424
Wireless bandwidth consumption used for DNS queries is therefore reduced 1426

FIG. 14

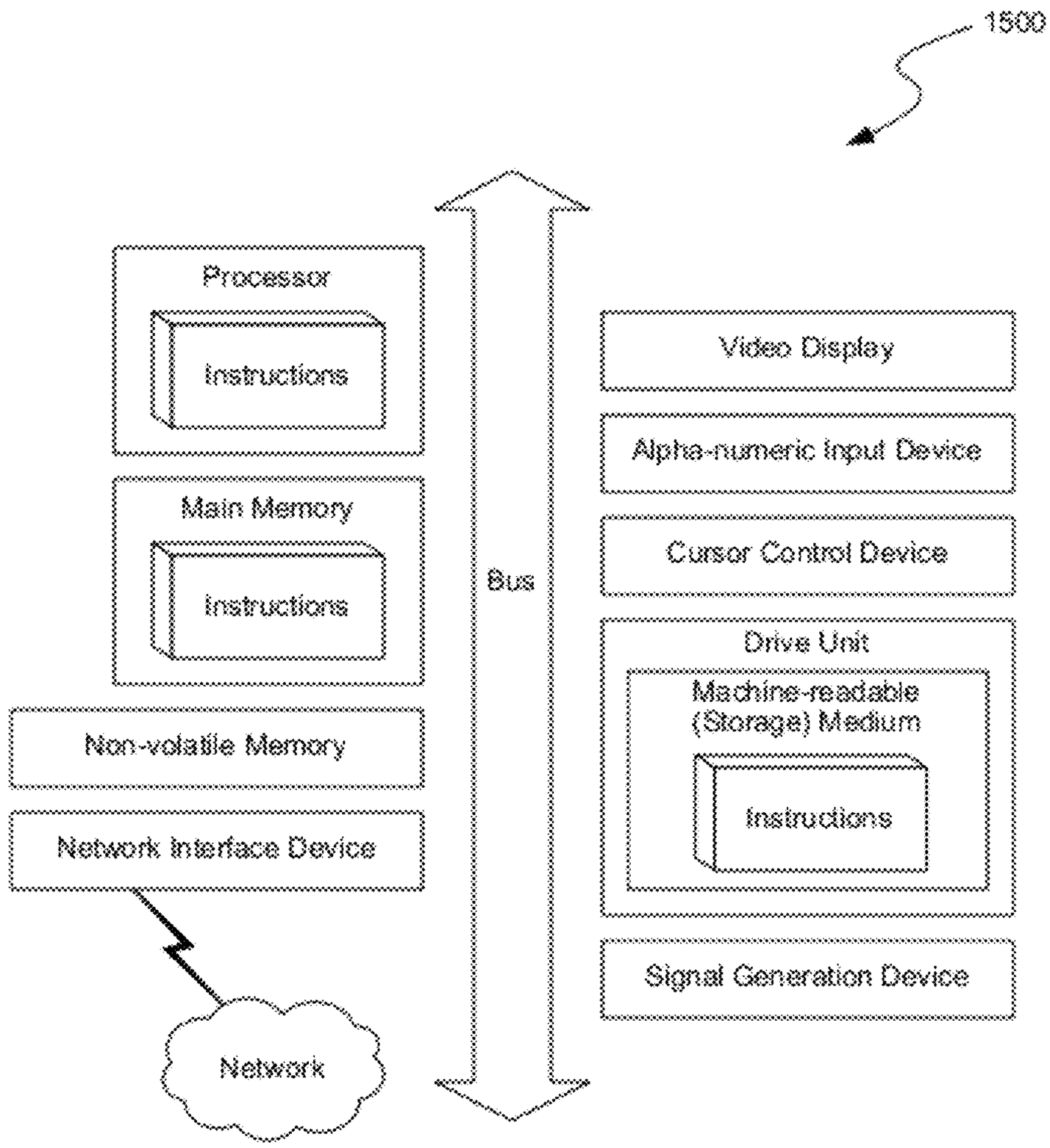

1500
Processor
Instructions
Main Memory
Instructions
Non-volatile Memory
Network Interface Device
Network
Bus
Video Display
Alpha-numeric Input Device
Cursor Control Device
Drive Unit
Machine-readable (Storage) Medium
Instructions
Signal Generation Device

FIG. 15

SYSTEM AND METHOD FOR REDUCTION OF MOBILE NETWORK TRAFFIC USED FOR DOMAIN NAME SYSTEM (DNS) QUERIES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application No. 61/430,828 entitled "DOMAIN NAME SYSTEM WITH NETWORK TRAFFIC HARMONIZATION," which was filed on Jan. 7, 2011, U.S. Provisional Patent Application No. 61/533,007 entitled "DISTRIBUTED CACHING IN A WIRELESS NETWORK OF CONTENT DELIVERED FOR A MOBILE APPLICATION OVER A LONG-HELD REQUEST," which was filed on Sep. 9, 2011, U.S. Provisional Patent Application No. 61/533,021 entitled "APPLICATION AND NETWORK-BASED LONG POLL REQUEST DETECTION AND CACHEABILITY ASSESSMENT THEREFOR," which was filed on Sep. 9, 2011, the contents of which are all incorporated by reference herein.

This application is related to U.S. patent application Ser. No. 13/176,537 entitled "DISTRIBUTED CACHING AND RESOURCE AND MOBILE NETWORK TRAFFIC MANAGEMENT," which was filed on Jul. 5, 2011, the contents of which are herein incorporated by reference.

This application is related to U.S. patent application Ser. No. 13/274,265 entitled "Caching Adapted for Mobile Application Behavior and Network Conditions," which was filed on Oct. 14, 2011, the contents of which are herein incorporated by reference.

This application is related to U.S. patent application Ser. No. 13/274,501 entitled "Request and Response Characteristics based Adaptation of Distributed Caching in a Mobile Network," which was filed on Oct. 17, 2011, the contents of which are herein incorporated by reference.

BACKGROUND

Domain Name System (DNS) records typically expire from a DNS resolver's cache (typically managed by the device operating system) after a short time-to-live period specified in the record. The short time-to-live period allows web site administrators to change the IP addresses of their servers without worrying that users will try to contact the servers at the old addresses and receive an error. However, when such requests are made over time in a mobile network, the repeated queries consume mobile network resources and consume mobile device battery life.

This is a concern for mobile devices where resources are limited since the constant connections and disconnections increase the amount of signaling network traffic, which lowers the performance of the network overall, and puts the pressure on network operators and providers to increase network bandwidth and network access. To date, carriers have been able to invest in 4G and LTE networks to boost network capacity in hotspots. However, these solutions are only accessible in metropolitan areas and 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.

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. The issue has been exacerbated by the rapid increase of the popularity of applications with network-initiated functionalities, such as push email, news feeds, status updates, multimedia content sharing, multimedia content streaming, mobile games and other mobile applications, etc. Furthermore, mobile phones also rely on signaling to send and receive calls and SMS messages and sometimes the fundamental phone-based functions are forced to take a backseat to mobile applications and other mobile clients.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1**A 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.

FIG. **1**B 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.

FIG. **1**C illustrates an example diagram of the logical architecture of a distributed proxy and cache system.

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

FIG. **1**E illustrates a diagram of the example components on the server side of the distributed proxy and cache system.

FIG. **2**A 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.

FIG. **2**B depicts a block diagram illustrating a further example of components in the cache system shown in the example of FIG. **2**A which is capable of caching and adapting caching strategies for mobile application behavior and/or network conditions. Components capable of handing various types of polls including detecting long poll requests and managing caching of long polls are also illustrated.

FIG. **2**C 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. **2**A which is further capable of detecting cache defeat and perform caching of content addressed by identifiers intended to defeat cache.

FIG. **2**D depicts a block diagram illustrating examples of additional components in the local cache shown in the example of FIG. **2**A which is further capable of performing

mobile traffic categorization and policy implementation based on application behavior and/or user activity.

FIG. **3**A 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.

FIG. **3**B 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. **3**A which is capable of caching and adapting caching strategies for mobile application behavior and/or network conditions. Components capable of handing various types of polls including detecting long poll requests and managing caching of long polls are also illustrated.

FIG. **3**C depicts a block diagram illustrating another example of components in the proxy system shown in the example of FIG. **3**A which is further capable of managing and detecting cache defeating mechanisms and monitoring content sources.

FIG. **3**D depicts a block diagram illustrating examples of additional components in proxy server shown in the example of FIG. **3**A which is further capable of performing mobile traffic categorization and policy implementation based on application behavior and/or traffic priority.

FIG. **4** depicts a block diagram illustrating another 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, caching, and/or traffic management, further including a DNS harmonization component.

FIG. **5**A 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, further including a DNS harmonization engine.

FIG. **5**B depicts a block diagram illustrating the interaction of a DNS harmonization component on a mobile device with a harmonization server to cache DNS entries to reduce the network traffic used for DNS queries.

FIG. **6**A 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.

FIG. **6**B 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.

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.

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

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.

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.

FIG. **11** depicts an interaction and timing diagram showing caching and retrieval of DNS cache entries for use and management in relation to mobile application requests and server polls.

FIG. **12** depicts a flow chart illustrating an example process showing a decision flow to determine whether the wireless network needs to be accessed to resolve a DNS query from a mobile application.

FIG. **13** depicts a flow chart illustrating an example process for a proxy to monitor a DNS server for changes to DNS queries for cache maintenance.

FIG. **14** depicts a flow chart illustrating an example process for reducing wireless bandwidth consumption used for resolving DNS queries made at mobile devices.

FIG. **15** 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

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.

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.

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.

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.

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.

Embodiments of the present disclosure include systems and methods for the reduction of mobile network traffic used for domain name system (DNS) queries.

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-availability, applications, user behavior, location thus changing the way the network interacts with the device and the application.

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.

The disclosed innovation transparently determines such "freshness" by monitoring, analyzing, and applying rules (which may be heuristically determined) 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.

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.

Embodiments of the disclosed technology includes, 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.

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.

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.

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.

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.

Domain Name System with Network Traffic Harmonization

Domain Name System (DNS) records typically expire from a DNS resolver cache after a short time-to-live period. This permits site administrators to change the IP addresses of their servers without worrying that users will try to contact those servers at the old addresses. Over time repeated queries consume network resources and consume battery life. This is a concern for mobile devices making DNS queries over mobile networks where bandwidth resources are already congested and limited.

In one embodiment, records returned from a Domain Name System (DNS) server or host are cached in a manner which integrates with a local content caching system and in some instances, also the web-based service addressed in the DNS query. The caching and management thereof can be performed by a DNS harmonization component in a local proxy on a mobile device (e.g., DNS harmonization component **401** or **526** of FIG. **4** and FIG. **5**B, respectively). The enhanced caching maintains and uses the cached DNS entries (e.g., stored in a local cache or harmonization cache **526** of FIG. **5**B) for a time period that is longer than the TTL typical of a DNS resolver. The increased TTL allows for a reduction in the network traffic related to DNS queries. By employing distributed monitoring mechanisms of a DNS server or host (e.g., DNS system **508** of FIG. **5**B) by a remote server (e.g., the harmonization server **510** of FIG. **5**B), it can be ensured that mobile applications or clients on the mobile device have access to valid DNS records.

In one embodiment, the enhanced caching for DNS-related traffic reduction performs the following example steps:

In the TCP/IP model, an Internet connection utilizing a transmission control protocol (TCP) socket connection can specify a destination Internet protocol (IP) address, for example, "69.63.189.11" where a browser or application will typically use a human-readable hostname identifier such as, "www.facebook.com". To convert from a hostname to an IP address, the application or client can query the device platform (e.g., the operating system) for a domain name resolver (e.g., DNS resolver) to resolve or otherwise determine and identify the hostname. For example:

1. Application to resolver: "I need to get an address for www.facebook.com".

2. Resolver to name server: "I've never heard of www.facebook.com, have you?"

3. Name server to resolver: "Yes, the address is 69.63.189.11".

4. Resolver to application: "Found it, you can open a socket to 69.63.189.11".

Note that the name server resides in the network and reaching it requires over-the-air traffic with radio access and battery consumption. To reduce this impact, a resolver typically includes a resolver DNS cache of recent DNS responses and the entries are typically maintained for a short duration. The resolver DNS cache is typically maintained by the operating system and the time to live of the cache entries are either specified by the operating system or configured via the operating system. In general, if an application tries to resolve a hostname twice within the time to live, it can receive a cached response from the resolver DNS cache instead of needing to go over the air. For example:

1. Application to resolver: "I need to get an address for www.facebook.com".

2. Resolver to application: "That's in my cache already, so I don't need to contact the Name server. You can open a socket to 69.63.189.11."

However, the time-to-live (TTL) of the DNS resolver cache (e.g., DNS resolver cache **522** shown in the example of FIG. **5**B) is typically relatively low (e.g., on the order of minutes, 5, 10, 15, 20, 30 min.). This low TTL is generally what causes frequent DNS queries at a mobile device to go over the cellular network. For example, if sequential requests are generated by local applications just above that setting (e.g., every 10, 15, 20, 25, 30, 45, or 60 minutes etc.), the system DNS resolver cache e.g., DNS resolver cache **522** of FIG. **5**B) will provide no benefit with respect to radio state change and reduction of traffic used for DNS queries sent over the wireless network. One embodiment of the disclosed technology utilizes, for example, a harmonization component (e.g., harmonization component **526** of FIG. **5**B) able to cache DNS records for longer periods of times (e.g., 30, 40, 50, 60, 75, 90, 120 minutes, etc.). The harmonization component is typically implemented locally on the mobile device and communicates with a remote server (e.g., harmonization server **510** of FIG. **5**B) which monitors a host or service to determine if and when a cached DNS record needs to be purged or updated.

In this manner, the DNS records maintained in the local cache on the local device can be used as long as the response from the actual DNS server or host has not changed as determined by the remote server or the harmonization server. This configuration and process allows the time period with which DNS records can be maintained by the harmonization component to be configurable and typically also dynamically adjustable, depending on factors including but not limited to, the predetermined lifetime per platform (by the OS), based on the requirement of a particular deployment, application settings, user preferences, application characteristics, device limitations, network operator settings, network operator policies or limitations, and/or application behavior and application characteristics.

As an example, consider a Facebook application (e.g., a mobile application on a mobile device) that polls a URL (e.g., "api.facebook.com") every 30 minutes: in this scenario, a polling task can be established at the Harmonization server to monitor responses to DNS queries. The initial DNS response for "api.facebook.com" may remain in the device's harmonization cache for the life of that polling task, or for another specified duration. The harmonization cache entry can be reference counted using a counter, so a second polling session referencing "api.facebook.com" will increment the counter (e.g., reference count). The counter can be decremented on invalidate or in response to receipt of stop poll commands from the harmonization server. Entries may expire from the harmonization cache when the counter reaches a certain number or the age in the cache reaches a threshold (e.g., 24 hours), whichever comes first, for example. Upon expiration, the DNS entries maybe automatically purged or marked as expired (e.g., in accordance with one or more metrics). Once marked as expired, the entries may be purged or purged upon verification that the entry is no longer valid/relevant.

Traffic Categorization and Policy

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.

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 (2) 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, (b3) immediate if the requesting application is in the foreground and active.

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.

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.

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.

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

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.).

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:

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

(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

(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).

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

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, 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.

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

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.

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.

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

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).

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

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.

FIG. **1**A 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 **120**A, promotional content server **120**B, or an e-coupon server **120**C 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 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 **120**A, promotional content servers **120**B, and/or e-Coupon servers **120**C as application servers or content providers are illustrated by way of example.

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 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.

The input mechanism on client devices **150** can include 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.

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.

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**).

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.

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.

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.

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. **1**B. 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**.

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).

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.

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).

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, WiFi, 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, 1×RTT, 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.

FIG. **1**B 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 **120**A, promotional content server **120**B, or an e-coupon server **120**C 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.

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 **120**A, promotional content servers **120**B, and/or e-Coupon servers **120**C as application servers or content providers are illustrated by way of example.

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).

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**.

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**.

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**.

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**-**3**.

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.

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.

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.

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 **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.

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.

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.

FIG. **1**C illustrates an example diagram showing an overview of the logical architecture of a distributed proxy and cache system.

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

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. **2**A-FIG. **2**D and FIG. **4**.

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. **3**A-FIG. **3**D and FIG. **5**A.

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.

FIG. **1**D illustrates an example diagram showing the logical architecture of the client side components **175** in a distributed proxy and cache system.

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:

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.

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**.

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.

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**.

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 scheduler **124** 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).

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.

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.

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.

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**).

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. For example, the proxy server **199** may be a DNS harmonization server or may perform the functions described for the harmonization server in conjunction with the description of FIG. **5**A-**5**B. 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.

FIG. **1**E illustrates a diagram of the example components on the server side **125** of the distributed proxy and cache system.

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:

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.

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).

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. 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.

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.

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.

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 all the necessary tools for producing reports and usage analytics that operators can use for analyzing application signaling and data consumption.

FIG. **2**A 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.

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., WiFi, cellular, Bluetooth, LAN, WAN, etc.).

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, WiFi 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.

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.

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.

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, etc.

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** and other mobile applications) to open TCP connections to a remote server (e.g., the server **100** in the examples of FIG. **1**A-**1**B and/or server proxy **125**/**325** shown in the examples of FIG. **1**B and FIG. **3**A). 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)).

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.

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.

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.).

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).

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.

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.

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.

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. **1**A-B and FIG. **3**A) 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.).

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. **1**A-B and FIG. **3**A) send the data that was buffered as a result of the previously decreased communication frequency.

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. **1**A-B and FIG. **3**A) 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.

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.

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.

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.

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.

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.

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.

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, geo-

graphic 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.

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.

TABLE I

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 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.

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.

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.

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.

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.

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.

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.

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.

In one embodiment, the local proxy **275** locally adjusts radio use on the device **250** by caching data in the cache **285**. When requests or transactions from the device **250** can be satisfied by content stored in the cache **285**, 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. **1**A and FIG. **3**A or a content provider/application server such as the server/provider **110** shown in the examples of FIG. **1**A and FIG. **1**B). As such, the local proxy **275** can use the local cache **285** 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.

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 **285** 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 **285**, 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.

If a valid response is not available, the local proxy **275** can query a remote proxy (e.g., the server proxy **325** of FIG. **3**A) 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. **1**B) 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.

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. **3**A) which forwards the data request to a content source (e.g., application server/content provider **110** of FIG. **1**A) 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. **3**A. 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.

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. **3**A).

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.

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.

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.

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.

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. **1**A) 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. **1**B or proxy server **325** of FIG. **3**A) 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.

As a result of the offloading of the polling, locally cached content stored in the local cache **285** 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. **3**A.

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.

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. **1**B) 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. **1**A).

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.

FIG. **2**B depicts a block diagram illustrating a further example of components in the cache system shown in the example of FIG. **2**A which is capable of caching and adapting caching strategies for mobile application behavior and/or network conditions.

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 **246***a,a* content predictor **246***b*, a request analyzer **246***c*, and/or a response analyzer **246***d*, and the cache or connect selection engine **249** includes a response scheduler **249***a*. 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.

The cache look-up engine **205** may further include an ID or URI filter **205***a*, the local cache invalidator **244** may further include a TTL manager **244***a*, and the poll schedule generator **247** may further include a schedule update engine **247***a* and/or a time adjustment engine **247***b*. 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 **238***a* having a response/request tracking engine **238***b*. The poll interval detector **238** may further include a long poll hunting detector **238***c*. The application profile generator **239** can further include a response delay interval tracker **239***a*.

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. **2**A. 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.)

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. **1**B) 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).

In one embodiment, the decision engine **246**, for example, via the request analyzer **246***c*, 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.

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.

In some instances, content from a given application server/content provider (e.g., the server/content provider **110** of FIG. **1**B) 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. **1**B and FIG. **2**A) 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.

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 **246***c*) 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).

Periodicity can be detected, by the decision engine **246** or the request analyzer **246***c*, 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 **246***a* 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.

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.

In one embodiment, the decision engine **246**, for example, via the response analyzer **246***d*, 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.

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.

In addition, the response characteristics information can include an associated status code of the response which can be identified by the response analyzer **246***d*. In some instances, HTTP responses with uncacheable status codes are typically not cached. The response analyzer **246***d* 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.

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 **246***d* 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**.

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 **246***d* 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.

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 **246***d*, 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.

The cache appropriateness decision engine **246** (e.g., the content predictor **246***b*) 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. **1**A-B). 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 **246***d*, 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.

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**.

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.

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, a timing diagram showing timing characteristics is illustrated in FIG. **8**), as detected by the long poll hunting detector **238***c* of the application behavior detector **236**.

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)

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).

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. **3**A for use in polling the content source (e.g., application server/service provider **310**)).

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.

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)

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 **238***c* 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.

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.

In one embodiment, the timing predictor **246***a* 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**).

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

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.

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.

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.

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.

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**).

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.

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.

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.).

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 **246***a* and/or the content predictor **246***b* 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**.

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 **246***a/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**).

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. **3**A, host **100** and proxy server **125** in the example of FIG. **1**A-B). 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.

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.

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. **1**B and FIG. **2**A, respectively) on the mobile device **250**.

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.

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 **244***a* 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.

A default time-to-live can be automatically used for all entries unless otherwise specified (e.g., by the TTL manager **244***a*), or each cache entry can be created with its individual TTL (e.g., determined by the TTL manager **244***a* 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. Examples of representations of a data model of a cache entry are illustrated in FIG. **24** and FIG. **25**.

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. **1**B and FIG. **3**A, 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. **1**B and FIG. **2**A, 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**.

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 **205***a*). 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.

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.

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

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

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.

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.

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.

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.

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.

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.

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. **1**A-B). 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.

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. **1**A-**1**B). 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.

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.

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 **238***a* 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 **249***a*.

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. **1**B and FIG. **3**A-C).

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).

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.

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.

One embodiment of the poll schedule generator **247** includes a schedule update engine **247***a* and/or a time adjustment engine **247***b*. The schedule update engine **247***a* 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**.

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 **247***a* 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.

In one embodiment, the time adjustment engine **247***b* can further optimize the poll schedule generated to monitor the application server/content source (**110** or **310**). For example, the time adjustment engine **247***b* 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.

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**.

To resolve this non-optimal result of serving the out-dated content once again before invalidating it, the time adjustment engine **247***b* can specify the time (t**0**) at which polling should begin in addition to the rate, where the specified initial time t**0** 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.

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 **238***b* of the long poll detector **238***a* of the poll interval detector **238**.

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').

The response delay time and idle time typical of a long poll are illustrated in the timing diagram shown below and also described further in detail with references to FIG. **17**A-B.

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

For example, the response/request tracking engine **238***b* 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. The relationship between requests can be seen in the timing diagrams of FIG. **17**A-B.

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

The response/request tracking engine **238***b* 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 **238***a* to determine whether requests generated by the application are long poll requests.

Note that in general, the first and second requests that are used by the response/request tracking engine **238***b* 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 is illustrated in the example of FIG. **8** and can be, for example, detected by the long poll hunting detector **238***c*. In other words, the requests tracked by the response/request tracking engine **238***b* and used for determining whether a given request is a long poll occurs after the long poll has settled (e.g., shown in **810** of FIG. **8** after the hunting mode **805** has completed).

In one embodiment, the long poll hunting detector **238***c* can identify or detect hunting mode, by identifying increasing request intervals (e.g., increasing delays). The long poll hunting detector **238***a* 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 **238***c* 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., 2 s, 5 s, 10 s, 15 s, 20 s, etc.) and can be used as a filter or threshold value.

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 **238***b*) 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 **238***a*. 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 **238***b* 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.

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 **238***b* 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., 1 s. 2 s. 3 s . . . etc.).

In one embodiment, the long poll detector **238***a* can compare the relative timings (e.g., determined by the tracker engine **238***b*) 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.

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 **238***b* and/or by the application profile generator **239** (e.g., the response delay interval tracker **239***a*). 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).

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. **3**A-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. **3**B.

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. **1**B and FIG. **3**A) 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). Referring to the timing diagram of a request/response timing sequence shown in the example of FIG. **17**A-B, the timing intervals 'RI', 'D', 'RT', and/or 'IT', or some statistical manipulation of the above values (e.g., average, standard deviation, etc.) may all or in part be sent to the proxy server.

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.

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.

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.

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. **1**B and FIG. **3**A). 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, as further described with reference to the interaction diagram of FIG. **4**B.

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.

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.

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.

FIG. **2**C 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. **2**A. 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.

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**.

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. **2**A.

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.

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**).

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 (e.g., a date and/or time format, as illustrated in parameters **702** shown in FIG. **7**). 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.

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 (e.g., as illustrated in parameters **752** shown in FIG. **7**) and/or time/dates using commonly employed formats for these parameters and performing pattern matching algorithms and tests.

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. For example, the request/response pairs shown in the examples of FIG. **7** illustrate that the responses (**704** and **754** in FIG. **7**) received are the same, even though the resource identifier includes a parameter (**702** and **752** in FIG. **7**) that changes with each request.

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).

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. **1**B) 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. **1**B) 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. **1**B and FIG. **2**A) 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.

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 any one or any combination of the criteria, and in any order, in identifying sources for which caching may be suitable.

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. **1**B and FIG. **2**A, 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. **1**B and FIG. **3**A, 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. **1**B and FIG. **2**A, 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.

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.

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. **2**A) 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**.

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**.

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. **1**B and FIG. **3**A) 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).

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. **1**B and FIG. **3**A) 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. **4**B.

In another embodiment, the proxy server (e.g., proxy server **125** or **325** shown in the examples of FIG. **1**B and FIG. **3**A) 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. **1**A and FIG. **1**B 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**.

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. **3**C).

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).

In one embodiment, the cached elements stored in the local cache **285** (shown in FIG. **2**A) 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**.

FIG. **2**D depicts a block diagram illustrating examples of additional components in the local proxy **275** shown in the example of FIG. **2**A which is further capable of performing mobile traffic categorization and policy implementation based on application behavior and/or user activity.

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

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 **241***c* and/or the traffic categorizer **241***d*. 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 **241***c*) since the traffic for a foreground application vs. a background application may be handled differently.

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.

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.

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 **241***a*) and/or is not time-critical (e.g., by the time criticality detection engine **214***b*).

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 **241***a*) and/or as being time critical/time sensitive (e.g., by the time criticality detection engine **241***b*).

The time criticality detection engine **241***b* 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.

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 **215***a*. 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 **215***b*. 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.

In addition, the user activity module **215** can also manage user expectations (e.g., via the user expectation manager **215***c* and/or in conjunction with the activity tracker **215** and/or the prediction engine **215***b*) 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.

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.

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.

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

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.

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.

FIG. **3**A 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.

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. **11**) able to receive or transmit signals to satisfy data requests over a network including any wired or wireless networks (e.g., WiFi, cellular, Bluetooth, etc.).

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**.

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**.

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, WiFi, 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.

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, WiFi interface, interfaces for vari-

ous generations of mobile communication standards including but not limited to 1G, 2G, 3G, 3.5G, 4G type networks such as LTE, WiMAX, etc.), Bluetooth, WiFi, 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.

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**.

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.

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.

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.

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**.

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**.

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**.

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.

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.

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. **3**A 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. **1**B), such as being managed by an application server/content provider **310**, a network service provider, or another third party.

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**.

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**.

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.

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.

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.

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**.

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**.

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. **1**B. 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.

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**.

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. **1**A). The proxy server can generate the keep-alive (heartbeat) messages independent of the operations of the local proxy on the mobile device.

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.

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.

FIG. **3**B 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. **3**A which is capable of caching and adapting caching strategies for application (e.g., mobile application) behavior and/or network conditions.

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 **358***a*, a long poll request detector/manager **358***b*, a schedule update engine **358***c*, and/or a time adjustment engine **358***d*. 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.

In one embodiment, the proxy server (e.g., the proxy server **125** or **325** of the examples of FIG. **1**B and FIG. **3**A) 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. **2**A-B. The content source (e.g., application server/content provider **110** of FIG. **1**B) 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**.

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 **358***a*. 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 **358***b*). 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.

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' shown in the example timing diagram of FIG. **17**A-B) 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.

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.

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).

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.

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. **1**B and FIG. **3**A, 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.

The metadata generator **303**, similar to the metadata generator **203** shown in the example of FIG. **2**B, 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. **2**B.

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 **246***d* of the local proxy shown in the example of FIG. **2**B. 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. Examples of response analysis procedures are also described in conjunction with the flow charts shown in the examples of FIG. **11**-**13**. 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**.

Furthermore, the schedule update engine **358***c* 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 **358***d* 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**.

FIG. **3**C 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. **3**A which is capable of managing and detecting cache defeating mechanisms and monitoring content sources.

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**.

In one embodiment, the proxy server (e.g., the proxy server **125** or **325** of the examples of FIG. **1**B and FIG. **3**A) 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. **1**B or **310** of FIG. **3**A) 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**.

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.

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.

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).

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.

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 **285**) 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 **285** (now invalid, out-dated, or otherwise determined to be irrelevant).

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. **1**B and FIG. **3**A, 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.

FIG. **3**D depicts a block diagram illustrating examples of additional components in proxy server **325** shown in the example of FIG. **3**A which is further capable of performing mobile traffic categorization and policy implementation based on application behavior and/or traffic priority.

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. **2**A-**2**D) or to an application server/content host (e.g., **110** of FIG. **1**A-**1**B). 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. **1**A-**1**B. 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.

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 **341***a*, a time criticality detection engine **341***b*, an application state categorizer **341***c*, and/or an application traffic categorizer **341***d*.

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.

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 **341***c* and/or the traffic categorizer **341***d*. 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 **341***c*) since the traffic for a foreground application vs. a background application may be handled differently to optimize network use.

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.

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.

The time criticality detection engine **341***b* 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.

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.

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 **341***c*), 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.

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.

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**).

FIG. **4** depicts a block diagram illustrating another 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, caching, and/or traffic management, further including a DNS harmonization component **401**.

The DNS harmonization component **401** and the DNS resolver **402** of the device operating system **204** operate in conjunction to provide cached responses to DNS queries of mobile applications **210** or **220**. The DNS harmonization component **401**, in one embodiment, maintains the cached responses to DNS queries longer than the DNS resolver **402** otherwise would thus reducing the instances where DNS queries would need to be placed over the air thus reducing mobile network traffic that would otherwise be consumed for DNS queries.

The DNS harmonization component **401** is able to intelligently maintain and serve cached responses to DNS queries for a longer period of time and ensure that they are still valid due to its interaction with a remote entity which monitors the application host/service provider for changed or new responses to given DNS queries (e.g., the server side proxy shown in the example of FIG. **5**A) as described below.

FIG. **5**A 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, further including a DNS harmonization engine **501**. The DNS harmonization engine **501** can monitor application hosts/servers for changes to DNS query responses and notify the client side (e.g., the DNS harmonization component **401**) to update the cached DNS query responses. The interaction of the client side DNS harmonization component and the server-side harmonization engine is further illustrated in the example of FIG. **5**B and described as follows:

FIG. **5**B depicts a block diagram illustrating the interaction of the DNS harmonization component **526** on a mobile device **502** with a harmonization server **510** to cache DNS entries to

reduce the network traffic used for DNS queries of applications **516** at the mobile device.

The architecture **500** for traffic reduction includes a mobile device **502** connected to a wireless network **504** by a radio **506**. The wireless network **504** is communicatively coupled to a domain name system server (DNS server) **508**. The communicative coupling of the wireless network **504** and the DNS server **508** can be facilitated by a harmonization server **510**.

The domain name system **508** may be a computer network (e.g., a public computer network such as the Internet) that incorporates one or more computers or servers configured to map domain names to numerical identifiers, for example, in accordance with one or more Internet Engineering Task Force (IETF) Request for Comments (RFC) documents including P. Mockapetris, "Domain Names—Concepts and Facilities", IETF RFC 1034, November 1987, P. Mockapetris, "Domain Names—Implementation and Specification", IETF RFC 1035, November 1987, R. Braden et al., "Requirements for Internet Hosts—Application and Support", IETF RFC 1123, October 1989, and/or Elz et al., "Clarifications to the DNS Specification", IETF RFC 2181, July 1997, each of which is incorporated herein by reference.

The mobile device **502** can include a system space **512** having computing resources reserved for an operating system of the mobile device **502**, and the user space **514** can include resources reserved for mobile applications **516** accessed by users of the mobile device **502**. For example, user application **516** can include a web browser or any mobile application/client that makes application programming interface (API) calls **518** to a DNS resolver **520**. In one embodiment, the DNS resolver **520** is a DNS resolver of a mobile device operating system. Typically, operating system DNS resolvers manage and maintain a DNS resolver cache **522**, which is configured at least to cache and store associations between domain names and network addresses determined from a DNS server **508**. In general, DNS resolvers **520** also generate DNS protocol messages **524**.

In accordance with embodiments of the present disclosure, the mobile device **502** further includes a DNS harmonization component **526** (e.g., such as the harmonization component **401** shown in the example of FIG. **4**) which can monitor and/or intercept outgoing DNS protocol messages **524** and associated responses. The DNS Harmonization component **526** can store DNS protocol request/response pairs in the harmonization cache **528**, and can match outgoing DNS protocol messages **524** to stored pairs. Should a match be found, the DNS Harmonization component **526** can generate a responsive DNS protocol message using the stored response of the matching pair, such that the query need not be sent over the radio **506** to resolve the DNS query.

If no match is found in the harmonization cache **528**, the DNS harmonization component **526** can instead, forward an outgoing DNS protocol message **524** to a corresponding domain name system (DNS) server **508** via the device radio **506**, and wireless network **504**, to resolve the DNS query. With respect to stored request/response pairs in the harmonization cache **528**, the DNS harmonization component **526** can instruct the harmonization server **510** to monitor the domain name system (DNS) server **508** for changes to corresponding domain name-network address associations ("address bindings").

The DNS harmonization component **526** can continue to serve the responses stored in the harmonization cache **528** typically until a change or invalidate is received from the harmonization server **510** (for a period of time that is longer than the DNS resolver **520** would serve responses in its cache **522**). Upon detection of a change, the harmonization server **510** may notify the DNS Harmonization component **526** to invalidate the corresponding harmonization cache **528** entry, so that it will be refreshed once the local DNS cache **522** entry becomes invalid (which occurs independently). Thus, cache entries stored in the harmonization cache **528** of the disclosed technology, are updated dynamically and changed when needed, rather than based on a pre-configured setting (e.g., of the device operating system) designed to capture quickest changes at any given host (e.g., DNS server). In this manner, different hosts can be monitored differently and DNS requests can be sent over the radio to access the wireless network **504** when needed, for any given query from any given application **516**. This is a lesser frequency than the situation when only the DNS resolver cache **522** of the DNS resolver **520** is used thus effectively conserving network resources and bandwidth consumption.

FIG. **6**A 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.

As shown in the distributed system interaction diagram in the example of FIG. **4**, 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. **4**) and the server side (e.g., server side **470** including the host server **485** and/or the optional caching proxy **475**).

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. **1**A-B).

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.

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.

The diagram of FIG. **6**A 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. **2**A-B or mobile device **450** of FIG. **4**) 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.

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.

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 from 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**.

FIG. **6**B 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**.

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 **454** directly from the server/provider **495**. If the data has been updated, the widget **455** on the mobile device **450** can refresh itself to reflect the update and waits for small period of time and initiates another data request to the server/provider **495**.

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.

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.

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**.

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 **462** the server **485** that the request should be monitored (e.g., steps **464** and **466**) for result changes prior to returning a result to the local proxy **465** or requesting widget **455**.

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 **468**, for which a locally cached response is available, is made by the widget **455** and intercepted at the local proxy **465**, the local proxy **465** can return the response **472** from the local cache without needing to establish a connection communication over the wireless network.

In addition, the server proxy performs the requests marked for monitoring **474** to determine whether the response **476** 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 **476** 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 **478**) and that the locally stored response on the client should be erased or replaced with a new response.

In this case, a subsequent data request **482** 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 **484**, 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.

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.

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.

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**)).

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**.

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.

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**.

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.

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**.

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.

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.

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).

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**.

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.

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**.

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**.

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.

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**.

FIG. **11** depicts an interaction and timing diagram **1100** showing caching and retrieval of DNS cache entries for use and management in relation to mobile application requests and server polls.

The DNS harmonization proxy on the client side proxy (e.g., local proxy **1165**) intercepts resolver requests (e.g., DNS protocol messages) that are intended to go over the wireless network to query a corresponding DNS server to resolve a DNS query. In one embodiment, the harmonization proxy on the local proxy **1165** functions similarly to a repeater, namely, forwarding the request including the DNS query over the network and subsequently also observing and recording the traffic. For example, at time 0m0s, once the DNS harmonization proxy of the local proxy **1165** has observed a request and response pair, it records them in the harmonization cache with a specified time-to-live (TTL) lifetime (e.g., 20, 25, 30, 40, 45, 60 minutes etc.). If the DNS Harmonization Proxy subsequently receives a similar request, for example, at time t=7m0s, it can provide the last successful response to satisfy the request instead of accessing the wireless network to forward the request over the air for resolution of a query which has previously been successfully resolved.

On the mobile device, the DNS harmonization proxy intercepts an outgoing DNS query for a hostname and check for matching responses in a Harmonization cache. Harmonization cache hits can be returned to a resolver (e.g., intercepted protocol messages can be replayed to the resolver). Cache misses may result in the outgoing DNS query being forwarded to the corresponding Domain Name Server for resolution, and the response can be stored into the DNS harmonization proxy's cache for future use.

When an application or client on the mobile device contacts the resolved IP address a related local content caching system can engage a network Harmonization server **1185** to monitor that content (e.g., at t=14m5s), and instruct the DNS harmonization proxy to retain the resolved IP address. For example, the harmonization server **1185** can poll the host to determine whether a response has changed.

The DNS harmonization proxy can now respond to queries for that cached address with confidence that the network harmonization server **1185** will advise or notify when the record becomes invalid (e.g., via a harmonization cache invalidation message). In one embodiment, resolution requests for that host name may now be serviced by the client-side DNS harmonization proxy **1165** resulting in reduced network and power usage. The polling continues (e.g., at t=21m5s through 70m5s) until the server **1185** detects new content in the polled response to the DNS query and notifies the harmonization proxy at the local client **1165**. The harmonization proxy at the local client **1165** receives the notification of invalid cache from the server **1185** and removes the invalid cache entry from the harmonization cache (e.g., at t=70 m10s).

The next time (e.g., at t=77 min 10 s) the harmonization proxy detects the same DNS query, the request is sent over the network (e.g., cellular, broadband, wifi, or other wireless networks) to the DNS server to resolve the query. The harmonization proxy **1165** again intercepts the response and is cached for subsequent use until notified of invalidation of the stored response.

FIG. **12** depicts a flow chart illustrating an example process showing a decision flow to determine whether the wireless network needs to be accessed to resolve a DNS query from a mobile application.

In process **1202**, an outgoing DNS query is detected at a mobile device. The outgoing DNS query may be initiated or caused to be generated by any background or user action performed on or via a mobile application or client on a device (e.g., a mobile or wireless device).

In process **1204**, cached DNS entries in a local cache on the mobile device are queried for a host name associated with the outgoing DNS query. In some instances, prior to querying the local cache for cached DNS entries, the resolver cache is queried for previously resolved cache entries. The resolver cache is typically maintained by the system's DNS resolver (e.g., by the operating system or the mobile platform) such as the DNS resolver **520**. The resolver cache which is queried (e.g., the DNS resolver cache **522**) can maintain DNS query responses for a short amount of time (e.g., a short time-to-live) that is typically shorter than the amount of time that the local cache (e.g., the harmonization cache **528** shown in the example of FIG. **5**B) maintains cached DNS query responses. In general, the resolved cache entries that are stored in the resolver cache (cache **522**) have a lifetime determined from domain name records associated with domain names and is shorter than lifetime of cached DNS entries in the local cache (e.g., the harmonization cache **528**).

Thus, in some instances, the resolver cache can be first queried to determine if any cached entries can resolve a given DNS query, and if not, since the local cache maintains cache entries for a longer period of the, the local cache can be queried in process **1206** to determine whether there is a matching cached DNS entry for the outgoing DNS query (e.g., the DNS protocol message **524** of FIG. **5**B).

If so, in process **1210**, the wireless network is not accessed and the matching cached DNS entry is provided as a response to the outgoing DNS query in process **1214**. If no matching entries exist, it is detected that the local cache is absent of a matching cached DNS entry for the outgoing query, in process **1216**. In process **1218**, the wireless network is thus accessed. In process **1220**, the outgoing DNS query is forwarded to a name server for resolution over the wireless network.

In process **1222**, a response provided for the outgoing DNS query by the name server is stored as a cached DNS entry in the local cache (or the harmonization cache **528** managed by the DNS harmonization component **526**). In one embodiment, the local cache is managed by a local proxy (which may be or may include the DNS harmonization component **526** (on the mobile device where the local proxy is able to communicate wirelessly with a remote entity (e.g. a remote

proxy) to facilitate traffic control and network resource management in a wireless network.

When caching DNS responses, the DNS harmonization component can simultaneously or in conjunction around the same time, set up a monitoring request with a remote entity (e.g., the remote server or remote proxy), requesting monitoring of the source/host for changes to a cached response for a given query. In general, the host which is able to resolve the DNS query is a remote host server for the mobile application or the mobile client which generated the DNS query. The remote entity can perform the monitoring of the DNS query responses on the mobile device or mobile clients' by polling the relevant server/hosts or name servers periodically using the DNS query, as further described in the flow chart of FIG. **13**.

In process **1224**, subsequent DNS queries that are the same as or similar to the outgoing DNS query are locally responded to, using the stored response. The matched pair can be stored for a specific lifetime, which can in some instances be in part or in whole platform dependent. The lifetime can also depend on the application parameters of the application generating the DNS query, device resources (storage, computing resource, user settings/preferences, account limitations, operator specifications, network conditions, and or user account type, etc.

FIG. **13** depicts a flow chart illustrating an example process for a proxy to monitor a DNS server for changes to DNS queries for cache maintenance at the harmonization cache (e.g., the harmonization cache **528** managed by the DNS harmonization component **526** of the example of FIG. **5**B).

In process **1302**, a polling task is sent to a proxy. The proxy is usually remote from the mobile device from which DNS queries are initiated (e.g., the harmonization server **510** of FIG. **5**B); in general, the proxy or proxy server is wirelessly coupled to the mobile device and are able to communicate with one another over the wireless network via radio access at the mobile device. The polling task typically includes instructions from the local proxy or the DNS harmonization component on the mobile device for the proxy to monitor a given application server/host or name server.

The application server/host or name server that is monitored can be specified in the polling task by the local proxy or harmonization component on the mobile device and typically corresponds to the server whose response to a DNS query at the mobile device is being cached. The polling task typically also includes the specific DNS query whose response is being locally cached by the harmonization component on the mobile device. In general, the proxy or proxy server can perform multiple polling tasks for one mobile device and/or perform polling tasks for multiple mobile devices. In process **1304**, the proxy polls a name server to monitor for changes in responses for the outgoing DNS query.

In process **1306**, the proxy detects changes for responses to the outgoing DNS query whose response has been cached at the mobile device. In response to detecting the change, in process **1308**, the proxy can notify the mobile device of the detected change or, in process **1310**, the proxy can notify the mobile device to invalidate the matching cached DNS for the outgoing DNS query. In response to the notification of the detected change or an invalidate indication from the proxy, the matching cached DNS entry is invalidated in the local cache in process **1312**. In process **1314**, the changed response for the outgoing DNS query is stored as a new DNS entry in the local cache for use in locally responding to subsequent DNS queries. In process **1316**, the new DNS entry for responding to the outgoing DNS query in locally responding to subsequent DNS queries

FIG. **14** depicts a flow chart illustrating an example process for reducing wireless bandwidth consumption used for resolving DNS queries made at mobile devices.

In process **1402**, a mobile application or mobile client on the mobile device sends a DNS query. In process **1404**, a local resolver cache of an operating system of the mobile device is queried for a matching response for the DNS query. In process **1406**, no matching entry is detected in the local resolver cache. In process **1408**, the DNS query is sent over a wireless network to a server. In process **1410**, the server communicates with a host able to resolve a DNS query. In process **1412**, the mobile device receives a response to the DNS query. In process **1414**, the mobile device stores the response as a cached entry for a host name associated with the DNS query.

In process **1416**, the cached entry stored on the mobile device is used as the host name. In process **1418**, the mobile device sends a polling task to the server for the server to monitor the host for changes in responses to the DNS query. In process **1420**, the cached entry stored on the mobile device for the DNS query is invalidated if the server identifies the change in the response to the DNS query In process **1422**, the mobile device stores a response provided for the DNS query by the server as an updated cached entry. In process **1424**, subsequent DNS queries are responded to without accessing the wireless network such that wireless bandwidth consumption is reduced. In process **1426**, wireless bandwidth consumption used for DNS queries is reduced.

FIG. **15** 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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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, 916, 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, 916 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.

What is claimed is:

1. A method for reducing network traffic used for DNS queries in a wireless network, the method, comprising:

querying cached DNS entries in a local cache on a mobile device for a host name associated with an outgoing DNS query from the mobile device, wherein each cached DNS entry stores an association between a domain name and a network address; and

providing a matching cached DNS entry as a response to the outgoing DNS query without accessing the wireless network;

wherein the matching cached DNS entry is invalidated in the local cache when a proxy remote from the mobile device detects a response for the outgoing DNS query and notifies the mobile device to invalidate the matching locally cached DNS entry; and

wherein the matching cached DNS entry is also invalidated in the local cache when a proxy remote from the mobile device originates a DNS query, detects a changed DNS response to the proxy-originated DNS query, and, in response to detecting the changed DNS response for the proxy-originated DNS query, notifies the mobile device to invalidate the matching locally cached DNS entry,

wherein the proxy-originated DNS query is performed in response to sending, by the mobile device, a polling task to the proxy that instructs the proxy to poll a name server to monitor for changes in responses to the proxy-originated DNS query,

wherein the polling task includes timing data such that the proxy-originated DNS query occurs before the mobile device performs a corresponding query to its local cache,

wherein the timing data is determined based on the freshness of mobile content.

2. The method of claim **1**, further comprising, in the absence of the matching cached DNS entry for the outgoing DNS query, forwarding the outgoing DNS query to a name server for resolution over the wireless network.

3. The method of claim **2**, further comprising, storing a response provided for the outgoing DNS query by the name server as a cached DNS entry in the local cache for use in locally responding to subsequent DNS queries that are the same as or similar to the outgoing DNS query.

4. The method of claim **1**, further comprising, querying resolved cache entries stored in a resolver cache for a response to the outgoing DNS query prior to querying the local cache.

5. The method of claim **2** wherein, the resolved cache entries have a lifetime determined from domain name records associated with domain names.

6. The method of claim **2**, wherein, lifetime of resolved cache entries stored in the resolver cache is shorter than lifetime of cached DNS entries in the local cache.

7. The method of claim **1**, wherein, the proxy is wirelessly coupled to the mobile device.

8. The method of claim **1**, wherein the timing data is determined such that the response to the proxy-originated query occurs enough before the mobile device performs the corresponding query to its local cache to allow the remote proxy to notify the mobile device to invalidate the matching locally cached DNS entry before the mobile device performs the corresponding query.

9. The method of claim **1**, further comprising, locally storing a matched pair of the outgoing DNS query and the matched DNS entry for locally responding to subsequent queries similar to or same as the outgoing DNS query.

10. The method of claim **2**, wherein, the matched pair is stored for a lifetime.

11. The method of claim **10**, wherein, the life time is platform-dependent.

12. The method of claim **1**, wherein, the outgoing DNS query is initiated by a mobile application accessed on the mobile device.

13. The method of claim **1**, further comprising, in the absence of a matching cache DNS entry for the DNS query in the local cache, forwarding the DNS query to a host server over the wireless network for resolution.

14. The method of claim **13**, further comprising, storing a response provided for the outgoing DNS query by the host server as a cached DNS entry in the local cache for use in locally responding to subsequent DNS queries.

15. The method of claim **1** wherein the timing data is determined based on an interval with which outgoing DNS queries are observed to be sent from the mobile device.

16. The method of claim **1**, wherein the freshness of mobile content is calculated based on the current age of the mobile content and a time-to-live (TTL) of the mobile content.

17. The method of claim **1**, wherein, a given cache entry in the local resolver cache has a shorter life time than the same given cache entry in the local cache.

18. A system for reducing wireless bandwidth consumption for performing DNS queries in a wireless network, the system, comprising:

a server which is able to communicate with a host able to resolve a DNS query, the DNS query being generated at

a mobile device that includes a local cache for caching DNS entries, wherein each cached DNS entry stores an association between a domain name and a network address and wherein the local cache provides a matching cached DNS entry as a response to the DNS query without accessing the wireless network;

wherein, the server is able to identify a change in a response to the DNS query from the host and in response invalidates the matching locally cached DNS entry by providing the change in the response to the mobile device; and

wherein the server also invalidates the matching cached DNS entry in the local cache in response to detecting, by the server, a changed DNS response from the host to a server-originated DNS query originated by the server of the mobile device,

wherein a proxy-originated DNS query is performed in response to sending, by the mobile device, a polling task to the proxy that instructs a proxy to poll a name server to monitor for changes in responses to the proxy-originated DNS query,

wherein the polling task includes timing data such that the proxy-originated DNS query occurs before the mobile device performs a corresponding query to its local cache,

wherein the timing data is determined based on the freshness of mobile content.

19. The system of claim **18**, wherein, the DNS query is generated by a mobile application or mobile client on the mobile device; wherein the host which is able to resolve the DNS query is a remote host server for the mobile application or the mobile client.

20. The system of claim **18**, wherein, mobile device stores a cached entry for a host name associated with the DNS query.

21. The system of claim **19**, wherein, the cached entry stored on the mobile device is used as the host name when a local resolver cache of an operating system of the mobile device does not include a matching response for the DNS query.

22. The system of claim **18**, wherein, in absence of a matching cache entry for the DNS query on the mobile device, the server sends the DNS query over the wireless network to the host for resolution.

23. The system of claim **21**, wherein, the mobile device stores a response provided for the DNS query by the server as an updated cached entry for use in responding to subsequent DNS queries without accessing the wireless network such that wireless bandwidth consumption is reduced.

24. A system for caching responses to DNS queries initiated by a mobile client, the system, comprising:

hardware for querying cached DNS entries stored in a local cache of the mobile client for a host name associated with a DNS query from the mobile client, when a local resolver cache does not contain a matching response for the DNS query, wherein each cached DNS entry stores an association between a domain name and a network address;

hardware for determining whether the cached DNS entries includes a match for the DNS query;

hardware for providing a matching cached DNS entry as a response to the DNS query without accessing a wireless network, in response to determining availability;

hardware for forwarding the DNS query over the wireless network for resolution, in response to determining absence of a matching cached DNS entry;

hardware for storing a response received over the wireless network for the DNS query as a cached DNS entry in the local cache for use in locally responding to subsequent DNS queries;

hardware for invaliding the matching cached DNS entry in the local cache when a changed response for the DNS query is detected by a proxy remote from the local device in response to a DNS query that originated from the mobile client and also in response to a proxy-originated DNS query that was originated of the mobile client,

wherein the proxy-originated DNS query is performed in response to sending, by the mobile device, a polling task to the proxy that instructs the proxy to poll a name server to monitor for changes in responses to the proxy-originated DNS query,

wherein the polling task includes timing data such that the proxy-originated DNS query occurs before the mobile device performs a corresponding query to its local cache, and

wherein the timing data is determined based on the freshness of mobile content.

* * * * *