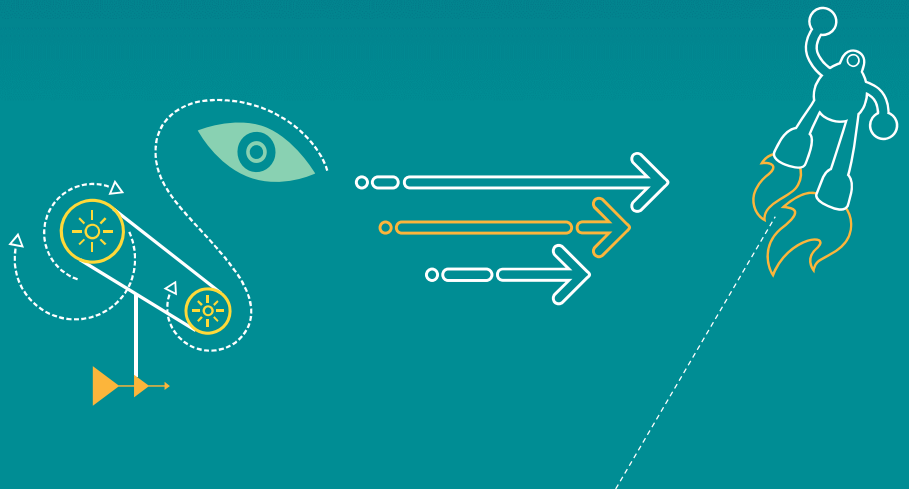
MSM8998/MSM8976/SDM660/SDM630 Data Power Manager (DPM) Overview

Qualcomm Technologies, Inc.



80-P2484-28 G

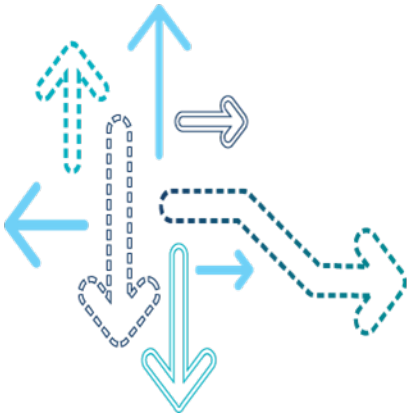
Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	April 2016	Initial release
B	May 2016	Added slides 52-54
C	August 2016	Added slides 48-57
D	October 2016	Updated the document title
E	November 2016	Updated slides 52-54
F	January 2017	Updated the document title to include SDM660
G	March 2017	• Added slide 25 • Updated the document title to include SDM630 • Updated slide 23

Contents

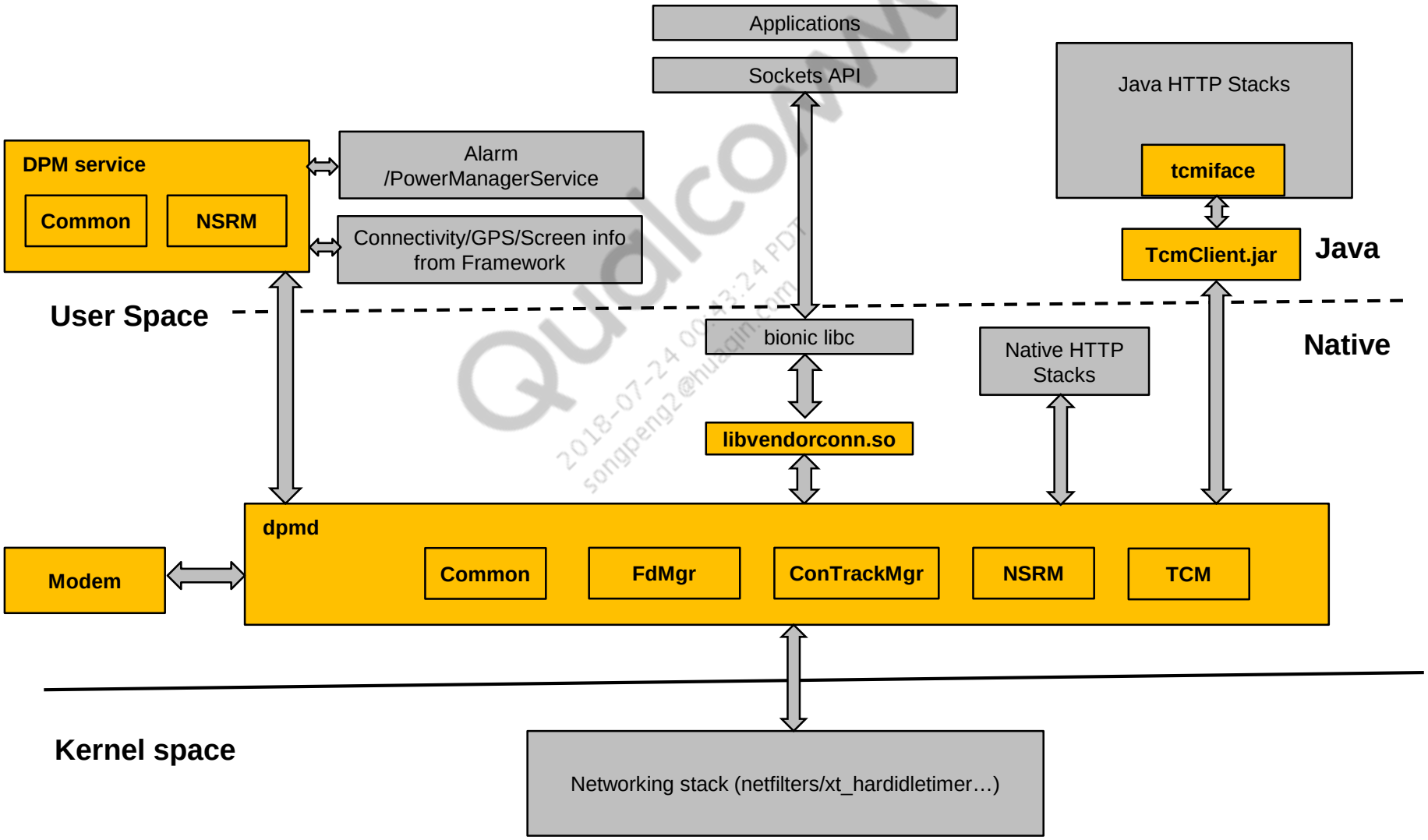
Introduction



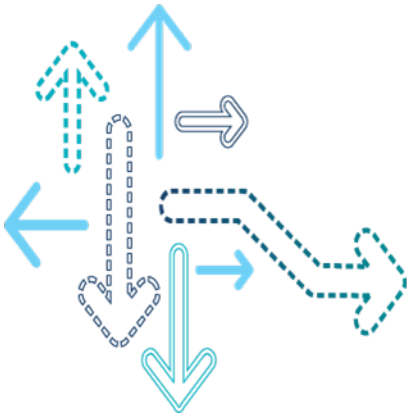
Data Power Manager

- Features
  - Fast Dormancy (FD)
  - Connection Tracking (CT) – For modem filtering and modem FD
  - Network Socket Request Manager (NSRM)
  - TCP Connection Manager (TCM)
- Persist property to control enable or disable
  - All DPM features are enabled by default.
    - Creates and configures persist.dpm.feature to disable specific features.
    - FD enable or disable – 0x00000001 (first bit 2 – LSB)
    - CT enable or disable – 0x00000002 (second bit)
    - NSRM 2.0 enable or disable – 0x00000004 (third bit)
    - TCM enable or disable – 0x00000008 (fourth bit)

High-Level Architecture



Fast Dormancy



Fast Dormancy Issue and Solution

- Issue
  - In some networks, the network inactivity timer is high and the RF chain on the UE stays active even after data inactivity. This occurs even in scenarios where the user is not actively using the phone, which costs power.
- Solution
  - Force the cellular data call to dormant, if there is no data activity (interface is idle) for longer than the configured amount of time.
  - Idle timer value depends on –
    - Screen state
      - ON – x sec
      - OFF – y sec
    - Tethering state (RNDIS QCMobileAP only)
      - ON (a cellular interface) – z sec, irrespective of the screen state
  - When tethering is active, the tethering timer is used irrespective of the screen state.

Fast Dormancy Configuration File

- Text file with TAG:value
- File
  - dpm_fd_screen_on_idle_timer_value: x
    - Amount of time for which the interface must be monitored for idleness when the screen is ON.
  - dpm_fd_screen_off_idle_timer_value: y
    - Amount of time for which the interface must be monitored for idleness when the screen is OFF.
  - dpm_fd_tethering_idle_timer_value: z
    - Amount of time for which the interface must be monitored for idleness when tethering is ON.
  - dpm_fd_enable_network_mask: mask
    - Bitmask stating for which networks' FD is to be enabled.
- Recommended configuration example
  - dpm_fd_screen_on_idle_timer_value:15
  - dpm_fd_screen_off_idle_timer_value:3
  - dpm_fd_tethering_on_idle_timer_value:15
  - dpm_fd_enable_networks_mask:0x28708
    - #Default 101000011100001000 (see Slide 11 for the bit selection of each network type).

Network Types for Selected Bitmask Value

- #Fast dormancy and TCM is configured for a network type.
  - #NETWORK_TYPE_UNKNOWN = 0
  - #NETWORK_TYPE_GPRS = 1
  - #NETWORK_TYPE_EDGE = 2
  - #NETWORK_TYPE_UMTS = 3
  - #NETWORK_TYPE_CDMA = 4
  - #NETWORK_TYPE_EVDO_0 = 5
  - #NETWORK_TYPE_EVDO_A = 6
  - #NETWORK_TYPE_1xRTT = 7
  - #NETWORK_TYPE_HSDPA = 8
  - #NETWORK_TYPE_HSUPA = 9
  - #NETWORK_TYPE_HSPA = 10
  - #NETWORK_TYPE_IDEN = 11
  - #NETWORK_TYPE_EVDO_B = 12
  - #NETWORK_TYPE_LTE = 13
  - #NETWORK_TYPE_EHRPD = 14
  - #NETWORK_TYPE_HSPAP = 15
  - #NETWORK_TYPE_GSM = 16
  - #NETWORK_TYPE_TD_SCDMA = 17
  - #NETWORK_TYPE_IWLAN = 18
- Example
  - Selection of UMTS – 0x8 (0000 1000) with third selected bit

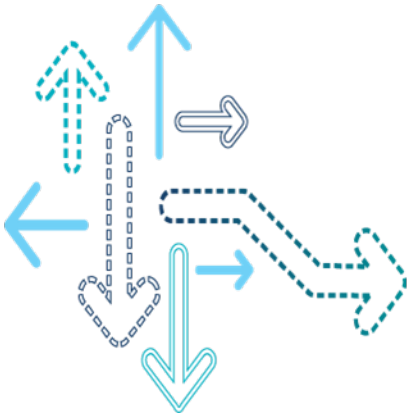
Recommended Configuration

- The default recommended configuration file is present in one for the following:
  - /system/etc/dpm/dpm.conf (targets supporting TCM)
- OEMs to push their configuration file to one for the following:
  - /data/dpm/dpm.conf (targets supporting TCM)
- Configuration parser logic is as follows:
  - If a valid file is present in /data, it is used. Otherwise, the parser falls back to the file in /system. For some reason if the file in that location is corrupted and parsing fails, the parser uses the software hardcoded recommended values.

How to Enable or Disable FD?

- To enable FD:
  1. adb root
  2. adb wait-for-device
  3. adb remount
  4. adb setprop persist.dpm.feature 1
     - This enables only the FD feature by changing the currently enabled features. To avoid disturbing an already enabled DPM feature, read the property value, set the LSB bit to 1 on top of it, and then set the property.
     - To update the configuration, do one of the following –
       - adb push dpm.conf /data/dpm/dpm.conf
  5. adb reboot
- To disable FD:
  1. adb root
  2. adb wait-for-device
  3. adb setprop persist.dpm.feature 0
     - This disables all the DPM features. To avoid disturbing any other DPM feature and only disable FD, read the property first, set the LSB bit to 0 on top of it, and then set the property.

Connection Tracking



Connection Tracking for Modem Filtering and Modem FD

- Issue
  - In some networks, there are spurious packets that unnecessarily wake up the modem and apps processor, which costs power.
- Solution
  - Track and report the active connections on the apps processor to the modem when the device goes idle (dormant).
    - Modem filters spurious download packets and does not wake up the apps processor unnecessarily.
    - On spurious packet reception, the modem aggressively goes dormant.
  - Save power in networks with spurious packets.
    - Less apps processor wake time
    - More modem RRC dormant time

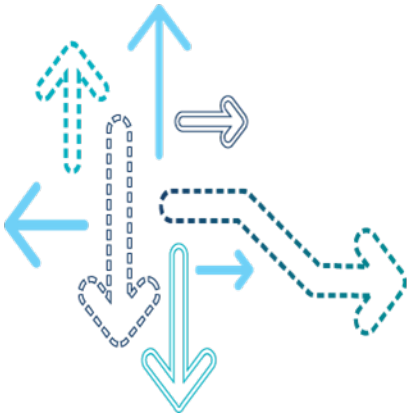
Modem Support and Configuration

- The CT feature works with the following modem support –
  - Modem performs filtering and drops packets for mismatched filters if the apps processor side installs a filter.
  - Modem triggers FD if the network has established a RAB for data transfer and there are no activity for 6 sec (configurable in EFS with TIMER_VALUE_1).
- Enable or disable modem FD and filtering, and configure FD timer values
  - To enable or disable the FD feature, push fast_dormancy.txt to EFS.
    - EFS file location – /ds/qmi/fast_dormancy.txt
    - EFS content
      - Write 1 to enable
      - Write 0 to disable
    - If the fast_dormancy.txt file is *not* present in EFS, FD is enabled by default.
  - To configure FD timer values in milliseconds, push fast_dormancy_config.txt to EFS.
    - EFS file location – /nv/item_files/modem/data/3gpp/fast_dormancy_config.txt
    - EFS content
      - TIMER_VALUE_1:xxxx; // FD timer with no packet drops
      - TIMER_VALUE_2:yyyy; // FD timer when packets dropped
    - TimerRange must be less than 10 sec && (TIMER_VALUE_1 > TIMER_VALUE_2).
    - If the fast_dormancy_config.txt file is *not* present, default values of 6000 and 1000 are used, respectively.

How to Enable or Disable CT?

- To enable CT:
  1. adb root
  2. adb wait-for-device
  3. adb remount
  4. adb setprop persist.dpm.feature 2
     - This enables only the CT feature by changing the currently enabled features. To not disturb an already enabled DPM feature, read the property value, set the second LSB bit to 1 on top of it, and then set the property.
  5. adb reboot
  6. Enable modem FD and filtering (see Slide 16)
- To disable CT:
  1. adb root
  2. adb wait-for-device
  3. adb setprop persist.dpm.feature 0
     - This disables all the DPM features. To not disturb any other DPM feature and only disable CT, read the property first, set the second LSB bit to 0 on top of it, and then set the property.
  4. adb reboot
  5. Disable modem FD and filtering (see Slide 16)

Network Socket Request Manager

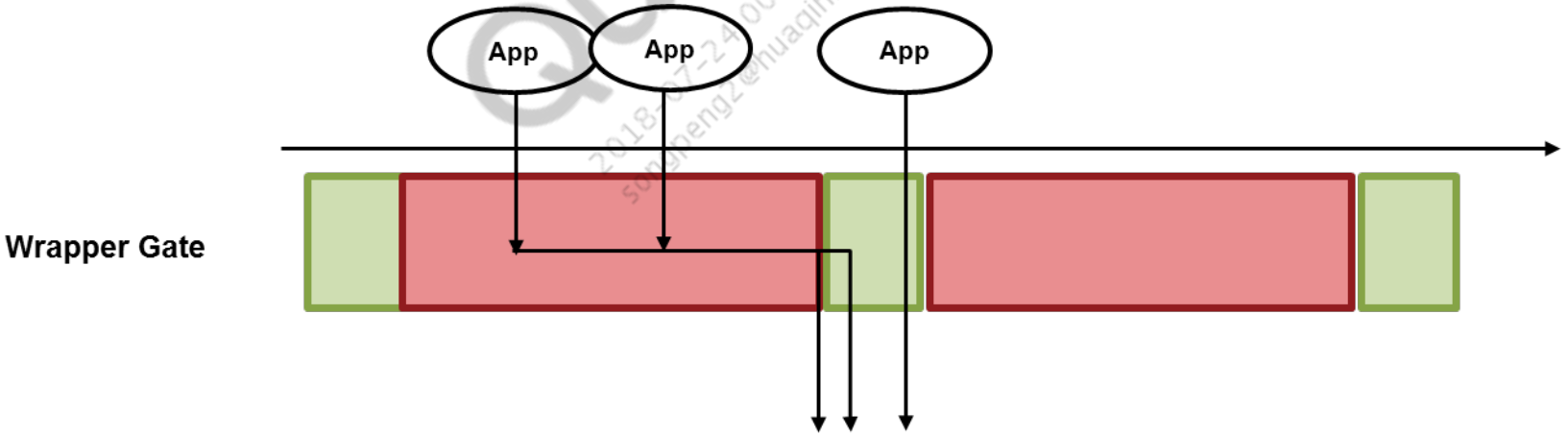


NSRM

- Issue
  - Applications such as Facebook and Twitter periodically pull data from the network and the requests are asynchronous with respect to time. This results in more RRC connections, i.e., more network signaling, which is directly related to battery consumption on the device.
- Solution
  - Synchronize the socket requests (DNS lookup, connect, or write) from the various applications when the device is in Background mode (RRC is dormant, no Wi-Fi, no streaming, etc.)
    - Reduces network signaling
    - Saves device power
    - Has no impact on user experience
  - The algorithm is based on the concept of the NSRM gate.

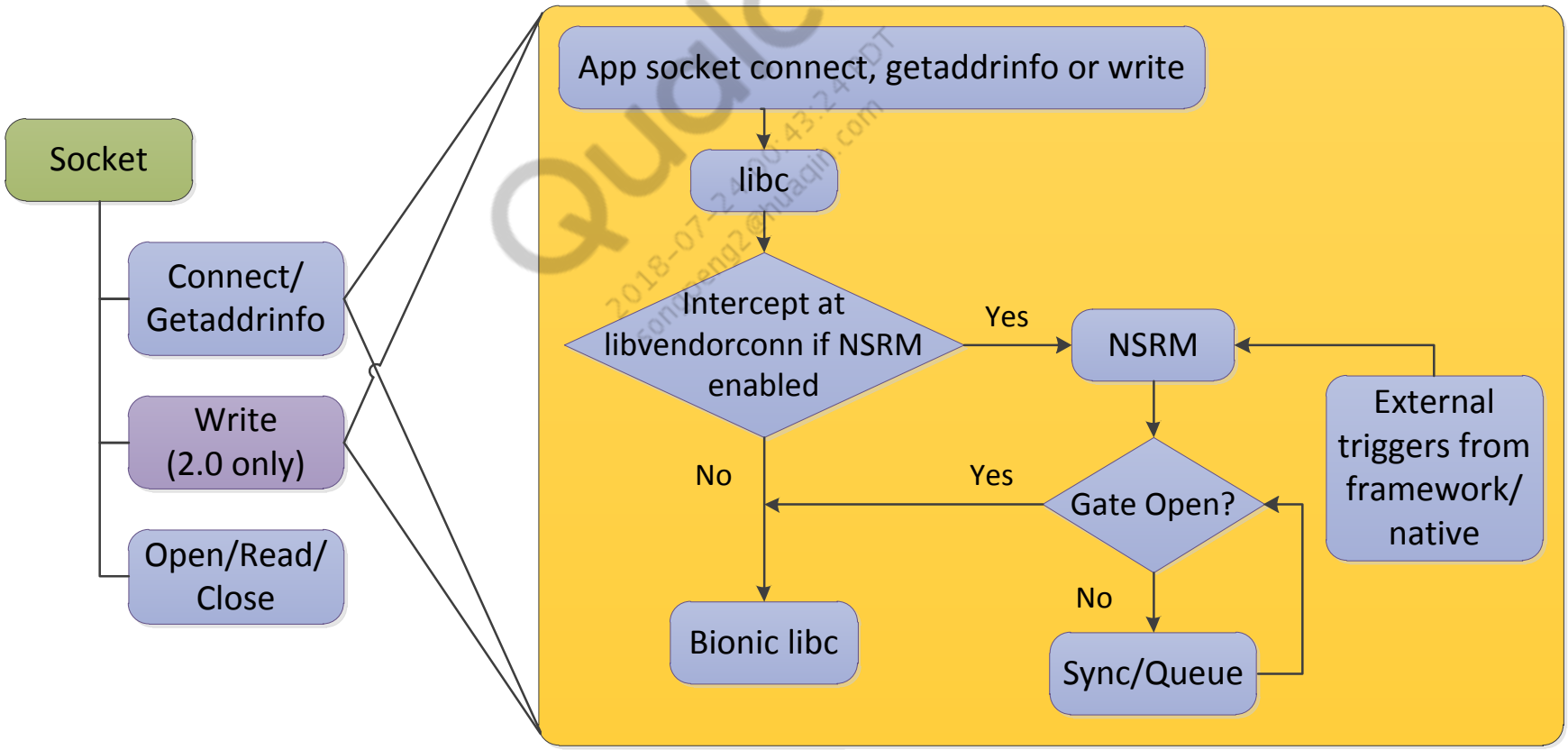
NSRM Gate

- When the gate is closed, selected socket calls are intercepted and synchronized or delayed until the gate is opened.
- The heart of the algorithm is the mechanism used to decide when to *open* or *close* its gate.
- Various system states and events affect the gate state.



High-Level Overview – NSRM 2.0

- NSRM 1.0 synchronizes TCP connect and DNS request.
- In addition, NSRM 2.0 synchronizes TCP write.



What NSRM Does and Does Not Synchronize

- Synchronizes
  - TCP connection setup
  - Calls that could result in DNS queries. For example, getaddrinfo and gethostbyname
  - TCP write on existing TCP connections
- Does not synchronize
  - Downlink-initiated traffic
  - TCP close
  - UDP socket calls other than DNS

NSRM Configuration File

- NSRM on bootup looks for a configuration file as follows –
  1. Primary location is /data/dpm/nsrm/.
  2. If it does not find the file at the primary location or the parsing of the file fails, NSRM looks for the file at /system/etc/dpm/nsrm.
  3. If it does not find any file at this alternate location or the parsing fails, NSRM falls back to default hardcoded values for the configuration parameters.
- In the builds released by Qualcomm Technologies Inc. (QTI), the recommended configuration file is present in /system/etc/dpm/nsrm; and /data/dpm/nsrm does not have any configuration file.
- For OEM applications, update the configuration file via the DPM functions.
- Refer to *Data Power Manager API Interface Specification* (80-NM328-61) for DPM functions and usage.
- NSRM on WLAN is supported. Few modifications are done in Configuration file to support them.

Example – NSRM 2.0 Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<NsrmPolicy>
 <Nsrm>
  <!--The version of the NSRM software-->
  <Version>2.0</Version>
  <!--Length of time in seconds to keep gate open each time -->
  <GateOpenTime>10</GateOpenTime>
  <!--Maximum time in seconds to wait before forcing gate open for connect or getaddrinfo -->
  <GateSyncSocketSetupTime>1200</GateSyncSocketSetupTime>
  <!--Maximum time in seconds to wait before forcing gate open for write -->
  <GateSyncSocketWriteTime>600</GateSyncSocketWriteTime>
  <!--Time in seconds to wait before releasing the queued sockets when the emergency alert notification is received
EAQSRT stands for Emergency Alert Queued Socket Release Delay Time -->
  <EAQSRDT>60</EAQSRDT>
  <!--Mode to indicate how processes which share the same UID should be handled-->
  <SharedUIDMode>Conservative</SharedUIDMode>
  <!--List of applications for which to apply Nsrm -->
  <AppList Type="Exclusion">
     <AppName>com.facebook.katana</AppName>
   <AppName>com.aol.mobile.engadget</AppName>
   <AppName>bbc.mobile.news.ww</AppName>
   <AppName>com.google.android.gm</AppName>
  </AppList>
  <!--NTO values are configured in seconds GATE will OPEN seconds before expiration of network binding. Port "0" is
the default configuration, applies if no port match is found -->
  <MCC_MNC value="210456">
   <port value="8080" NTO="1200"/>
   <port value="56" NTO="1800"/>
   <port value="0" NTO="300"/>
  </MCC_MNC>
  <MCC_MNC value="Default">
   <port value="0" NTO="300"/>
  </MCC_MNC>
 </Nsrm>
</NsrmPolicy>
```


Example – NSRM 2.0 Configuration (with NSRM on WLAN Support)

```
<?xml version="1.0" encoding="UTF-8"?>
<NsrmPolicy>
 <Nsrm>
  <!--The version of the NSRM software-->
  <Version>2.0</Version>
  <!--Length of time in seconds to keep gate open each time it is opened
        valid values 1 to 24*60*60 seconds-->
  <GateOpenWwanTime>1</GateOpenWwanTime>
  <!--Length of time in seconds to keep gate open each time it is opened
        valid values 1 to 24*60*60 seconds-->
  <GateOpenWlanTime>10</GateOpenWlanTime>  <!--Maximum time in seconds to wait before forcing gate open for connect or
getaddrinfo -->
  ……
  ……
  ……
  <MCC_MNC value="Default">
   <port value="0" NTO="300"/>
  </MCC_MNC>
   <MCC_MNC value="Wlan">
        <port value="0" NTO="900"/>
   </MCC_MNC>
  ……
  ……
  ……

 <!-- Nsrm Wlan Enable . 0 Disable, 1 Enable -->
 <NsrmWlanEnable>0</NsrmWlanEnable>

 <!-- Nsrm Wlan 0 Aggressive, 1 Conservative -->
<NsrmWlanMode>1</NsrmWlanMode>

</Nsrm>
</NsrmPolicy>
```

Tssync, Twsync and Topen Timers

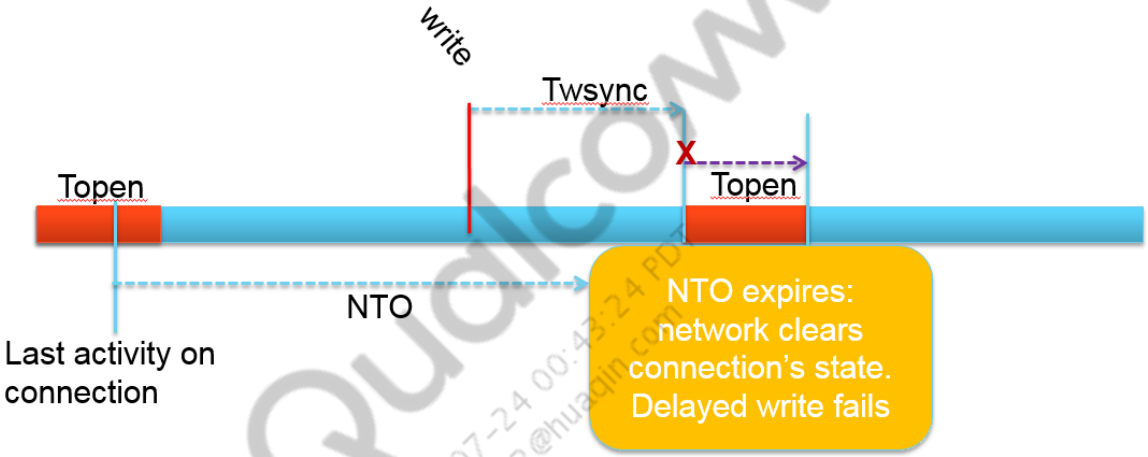
- These timers ensure that the applications are not blocked indefinitely.
- Tssync starts when the first connect or DNS request is captured after the gate is closed.
- Twsync starts when the first write call is captured after the gate is closed.
- The gate opens as soon as Tssync or Twsync expires, unless the gate opens earlier.
- Topen timer starts when the gate state changes to Open, and keeps the gate state Open until the timer expires. After the timer expires, if all the other conditions are met, the gate state changes to Close.
- All these timer values are configurable.

Tnto Timer

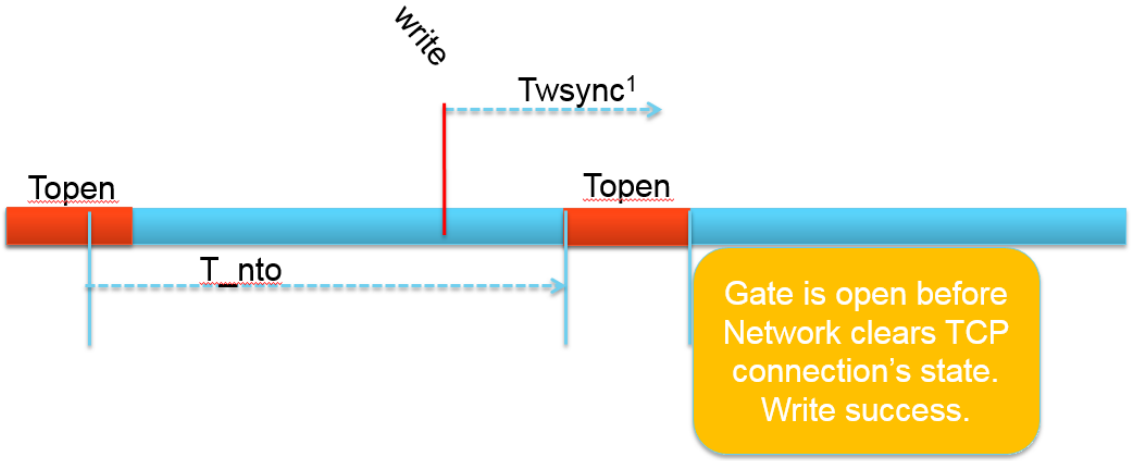
- Tnto ensures that NSRM does not synchronize write traffic beyond the inactive time used by the NAT and Firewalls.
  - Stateful middleboxes in the network, such as NAT or Firewalls, must maintain the state of each connection.
  - The state of the NAT or Firewall is erased after a certain amount of inactive time. When the state is erased, the TCP connection is unusable.
  - Tnto ensures that the synchronized write calls are released before the state is erased.
- There is one *Tnto timer* for each TCP connection.
- The gate opens as soon as *Tnto timer* expires, unless the gate opens earlier.
- *Tnto timer* restarts with its configured value when the TCP connection is used.

Example – Tnto

- Twsync delaying beyond network inactivity



- Opening the gate to avoid exceeding NTO



Tnto Value Configuration

- Tnto value is configured per the following –
  - Mobile operator identified by MCC/MNC
  - Destination port number
- If the current mobile operator is not present in the configuration file, use the values for default_mnc_mcc.
- If the current destination port is not present in the configuration file, use the value for default_port.
- Default_mnc_mcc
  - Default_port
  - NTO = 5 min
- Operator configures NTO for *partner operators.*
- Tnto value must be configured according to the actual timeout value in the operator's network for each port.

Example – Tnto Value Configuration

- Home operator has mccmnc = 123987 and the following NAT or Firewall inactivity timer configuration:
  - Port 80 – 20 min
  - Port 413 – 5 min
  - Default – 15 min
- Home operator partners with a neighbor operator to ensure optimal NSRM operation on the network. The neighbor operator has mccmnc = 465753 and the following port configuration:
  - Port 5228 – 35 min
  - Default – 13 min
- Configuration of the NAT and Firewall outside these two countries is unknown. As a precaution, NSRM is informed with a short inactive timeout (5 min). Alternatively, the delay of write() is turned OFF in those countries by setting a zero value, as shown in the example in Slide 31.

Example – Tnto Value Configuration (cont.)

- To set a relevant portion of NSRM configuration:

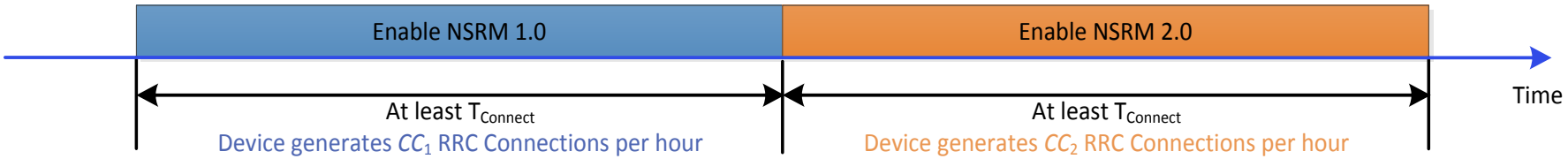
```
<MCC_MNC value="123987">
          <port value="80" NTO="1200"/>
          <port value="413" NTO="300"/>
          <port value="0" NTO="900"/>
</MCC_MNC>
<MCC_MNC value="465753">
          <port value="5228" NTO="2100"/>
          <port value="0" NTO="780"/>
</MCC_MNC>
<MCC_MNC value="Default">
          <port value="0" NTO="0"/>
</MCC_MNC>
```

Dynamically Enable NSRM 2.0 (DNSRM)

- Issue
  - NSRM 2.0 delays write() calls and sometimes causes an issue.
    - Some applications might send more traffic and trigger more RRC connections.
      - For example, in Whatsapp, if write() calls are delayed for more than 30 sec, the application closes the current socket and reconnects to a different server.
    - NSRM 2.0 should not be applied to this type of application.
  - A search over all applications is not done to find all the NSRM non-friendly applications.
- Solution
  - Detect dynamically if NSRM performs well on the UE.
  - Enable NSRM 2.0 dynamically per application.

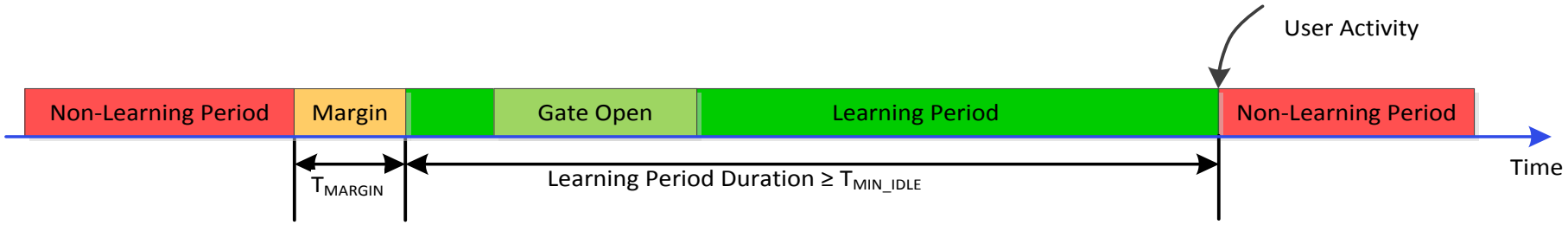
Dynamically Enable NSRM 2.0 – Overview

- Count the number of connect() calls for each application with NSRM 1.0 and with NSRM 2.0. Count for at least $T_{Connect}$ (240) minutes.
  1. If $CC_2 > BAD_{Thr} \times CC_1$, disable NSRM 2.0
  2. Else if $CC_2 \leq GOOD_{Thr} \times CC_1$, enable NSRM 2.0
  3. Else recheck
- Only count when the user is not using the device.
- Rerun the algorithm when one of the following occurs:
  - The application is updated.
  - 30 days since the last check.



Dynamically Enable NSRM 2.0 – When to Count?

- Do *not* count when the user is using the device; this is the non-learning period.
- Count only when the device is idle; this is the learning period.
  - Add a margin of T_{MARGIN} (60) seconds after the end of the non-learning period.
    - Allows the applications to cool down.
  - The learning period must be at least T_{MIN_IDLE} (5) minutes.
    - There might be no RRC connections during a short idle time, which might make the counting inaccurate.
- The application learning period is when the application is running and during a valid period.



Dynamically Enable NSRM 2.0 – Configurations

- DNsrmEnable – Enable and disable the DNSRM feature
  - If set to 0, no learning occurs.
- MinThr – If $\max\{CC_1, CC_2\} \leq Min_{Thr}$ for an application, enable write synchronization for that application
  - This means there is not much traffic from this application; enabling NSRM 2.0 is safe.
- GoodThr – If $CC_2 \leq GOOD_{Thr} \times CC_1$ for an application, enable write synchronization for that application.
- BadThr – If $CC_2 > BAD_{Thr} \times CC_1$ for an application, disable write synchronization for that application.
- $T_{Connect}$ – Minimum duration of the connection count
  - DNSRM counts connections for at least $T_{Connect}$ amount of time in both NSRM 1.0 and NSRM 2.0 modes before making the final decision.

Dynamically Enable NSRM 2.0 – Configurations (cont.)

- TMinIdle – Minimum duration of the valid learning period.
  - The length of any idle period must be longer than TMinIdle to become a valid learning period.
  - Idle periods shorter than TMinIdle are ignored; there is no connection counting during these short idle periods.
- TMargin – Margin from the learning period to the non-learning period.
  - Learning period starts after TMargin amount of time from the end of the non-learning period.
  - This margin gives some time for applications to cool down.
- Tquery – NSRM checks if an application is running every Tquery amount of time.
- TDecisionMade and TDmThreshold – NSRM re-evaluates the application. whose decision is made according to one of the following:
  - This application is updated *and* it is TDmThreshold seconds from the last time an evaluation was performed.
  - It is TDecisionMade seconds from the last time a check was performed.

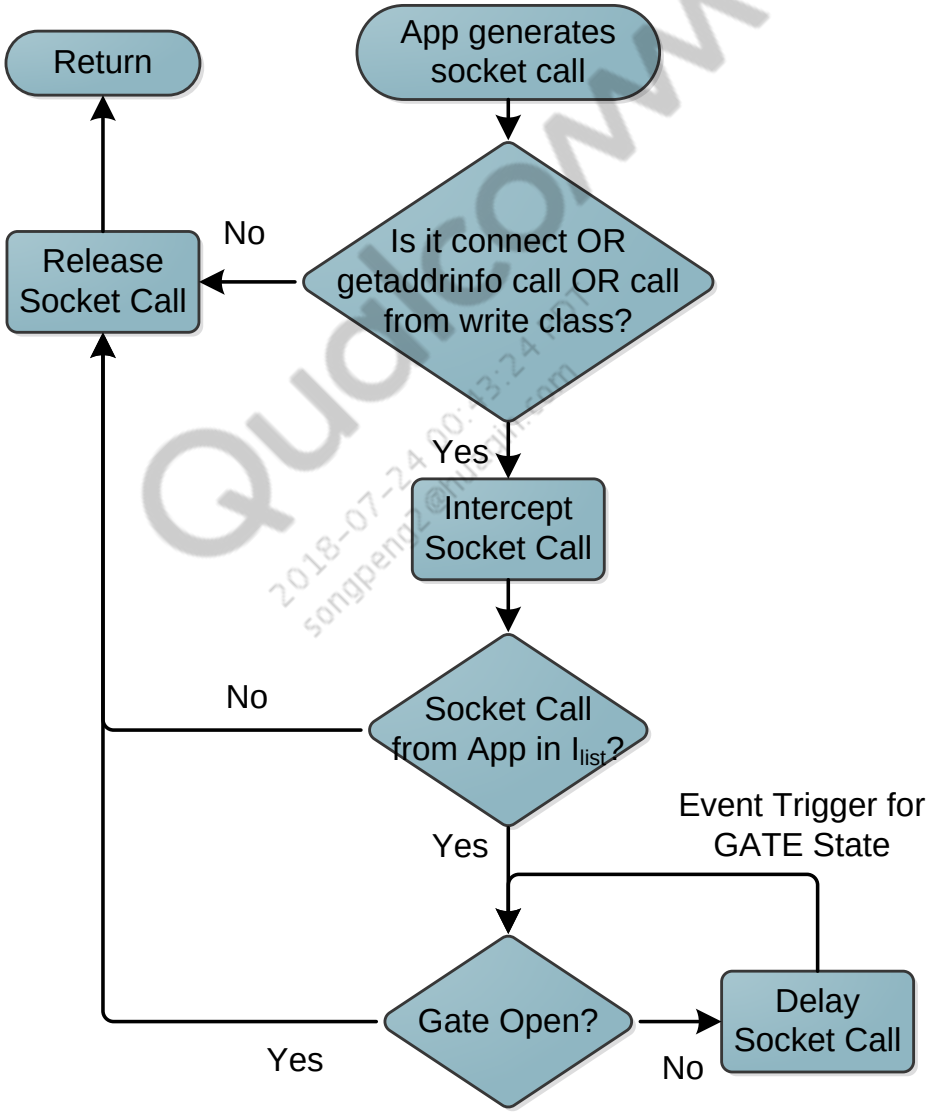
Emergency Alert

- The reception of an emergency alert triggers opening of the gate.
  - Emergency alerts are broadcast to all devices in an area.
  - If all these devices open the gate and release the synchronized sockets, there is a traffic explosion on the network.
  - Configuration is required to decide whether or not an emergency alert opens the gate.
- To avoid a traffic explosion, upon the reception of an emergency alert, the following occurs –
  1. A timer is started with a value selected randomly between 0 and EAQSRDT (the emergency alert queued socket release delay time).
  2. The gate opens and remains open while the timer is running.
  3. The socket calls that were queued prior to reception of the emergency alert are not released upon opening the gate.
  4. New socket calls are not queued (normal behavior because the gate is open).
  5. While the gate is open, when a new socket call originates from an application with UID = x, all the queued socket calls from the application with UID = x are released.
  6. When the timer expires, all the queued socket calls are released (this might lead to opening the gate due to a radio connection).
- EAQSRDT is configurable.

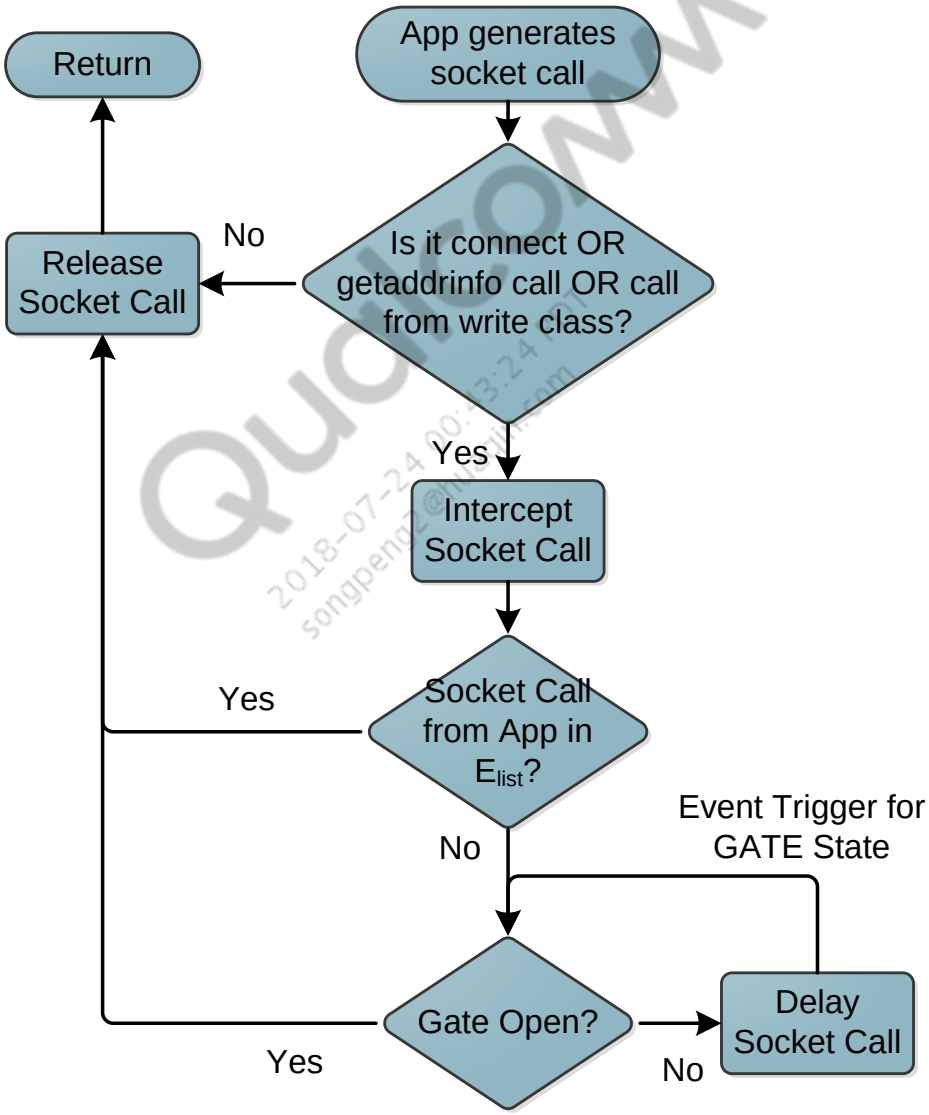
NSRM Modes of Operation

- NSRM operates in two modes:
  - Inclusion
  - Exclusion
- In Inclusion mode, only applications in an Inclusion List (ILIST) are subject to NSRM.
- In Exclusion mode, all applications except the ones in an Exclusion List (ELIST) are subject to NSRM.
- Each of the modes are either conservative or aggressive.
- These settings are specified in an operator-configurable .xml policy file (NsrmConfiguration.xml).

Inclusion Flowchart – Application is in ILIST



Exclusion Flowchart – Application is in ELIST



Aggressive vs. Conservative

- Android OS allows applications to share UIDs.
  - This causes ambiguity in certain scenarios about whether an application is to be subjected to NSRM. For example, application A is listed in the ILIST, but application B shares the same UID with application A. If a Socket Open request comes from this UID, should it be subject to NSRM?
- The following table shows how to decide whether a socket generated by an application is to be synced by NSRM.

	Inclusion	**Exclusion**
Conservative	All applications with the same UID must be in the **ILIST**	All applications with the same UID must be listed in the **ELIST**
Aggressive	Any application with the same UID listed in the **ILIST**	Any application with the same UID listed in the **ELIST**

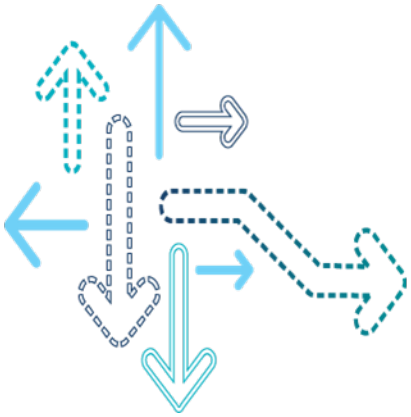
Aggressive vs. Conservative (cont.)

- Inclusion and conservative
  - All the applications on the device that share that UID must be listed in the ILIST for NSRM to begin.
  - If there is even one application that shares the same UID but is not listed in the ILIST, the socket is not to be synchronized and is allowed to proceed immediately.
- Inclusion and aggressive
  - If any application on the device that shares a UID is listed in the ILIST, the socket is synchronized.
- Exclusion and conservative
  - All the applications on the device that share the UID must be listed in the ELIST for that socket to be excluded from NSRM. That is, if the operator policy does not include all applications that share the UID, the socket is synchronized.
- Exclusion and aggressive
  - If any application on the device that shares the UID is listed in the ELIST, the socket is not synchronized.

How to Enable or Disable NSRM?

- To enable NSRM:
  1. adb root
  2. adb wait-for-device
  3. adb remount
  4. adb setprop persist.dpm.feature 4
     - This enables only the NSRM feature by changing the currently enabled features. To not disturb an already enabled DPM feature, read the property value, set the third LSB bit to 1 on top of it, and then set the property.
     - To update the configuration:
       adb push NsrmConfiguration.xml /data/dpm/nsrm/NsrmConfiguration.xml
     - To update the trigger mask (only for test):
       adb shell setprop persist.dpm.nsrm.bkg.evt <mask>
  5. adb reboot
- To disable NSRM:
  1. adb root
  2. adb wait-for-device
  3. adb setprop persist.dpm.feature 0
     - This disables all the DPM features. To not disturb any other DPM feature and only disable NSRM, read the property first, set the third LSB bit to 0 on top of it, and then set the property.
  4. adb reboot

TCP Connection Manager



TCP Connection Manager

- Issue
  - Android applications such as Twitter, Gmail, and Instagram do not close TCP sockets after the data exchange. Instead, they wait for the server to initiate closure of the TCP connections. After some inactivity, the server decides to close the TCP connection and sends a TCP FIN to the terminal or client. This results in additional RRC connections or network signaling, which impacts network capacity and UE battery life.
- Solution
  - Use a mechanism to close the idle TCP sockets on behalf of the application, without impacting the end-user experience, after the data transfer is complete.
  - Modify the HTTP stacks that ship with the Android platform so that the idle connections in the connection pools are closed before the radio goes dormant. This mainly requires modifying the standard HTTP stacks.

Example Configuration File – dpm.conf

```
#configuration parameters for DPM Fast Dormancy and TCM module.

#Configuration params for FD
#Idle timer value when SCREEN state is ON
dpm_fd_screen_on_idle_timer_value:15
#Idle timer value when SCREEN state is OFF
dpm_fd_screen_off_idle_timer_value:3
#Idle timer value when TETHERING is ON
#This takes precedence over SCREEN state
dpm_fd_tethering_on_idle_timer_value:15
#FastDormancy can be configured for a network type
#Default configuration 1010000111000001000
dpm_fd_enable_networks_mask:0x28708

#Configuration params for TCM
#Idle timer value when SCREEN state is ON
#min : 1s and max :256s
dpm_tcm_screen_on_idle_timer_value:5
#Idle timer value when SCREEN state is OFF
#min : 1s and max :256s
dpm_tcm_screen_off_idle_timer_value:1
#TCM can be configured for a network type
#Default configuration 1111111111111110
dpm_tcm_enable_networks_mask:0x7FFFE
```


Recommended Configuration

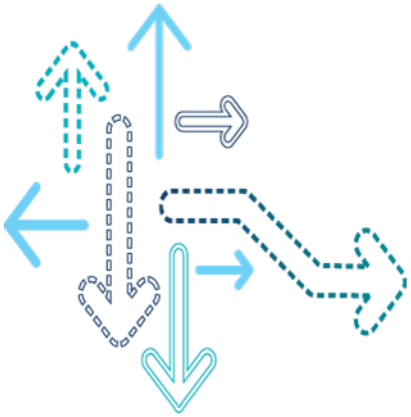
```
#Idle timer value in seconds when SCREEN state is ON
dpm_tcm_screen_on_idle_timer_value:5
#Idle timer value in seconds when SCREEN state is OFF
dpm_tcm_screen_off_idle_timer_value:1
#TCM can be configured for a network type; it's a bit mask
#Default configuration 1111111111111110 (Enabled everything)
dpm_tcm_enable_networks_mask:0x7FFFE
```

- The default recommended configuration file for DPM is present in /system/etc/dpm/dpm.conf.
- OEMs to push their configuration file for DPM to /data/dpm/dpm.conf
- Configuration parser logic is as follows:
  - If a valid file is present in /data, it is used. Otherwise, the parser falls back to the file in /system. If for some reason the file in that location is corrupted and parsing fails, the parser uses the software hardcoded recommended values.

How to Enable or Disable TCM?

- To enable TCM:
  1. adb root
  2. adb wait-for-device
  3. adb remount
  4. adb setprop persist.dpm.feature 8
     - This enables only the TCM feature by changing the currently enabled features. To not disturb an already enabled DPM feature, read the property value, set the fourth bit to 1 on top of it, and then set the property.
     - To update the configuration:
       adb push dpm.conf /data/dpm/dpm.conf
  5. adb reboot
- To disable TCM:
  1. adb root
  2. adb wait-for-device
  3. adb setprop persist.dpm.feature 0
     - This disables all the DPM features. To not disturb any other DPM feature and only disable TCM, read the property first, set the fourth bit to 0 on top of it, and then set the property.
  4. adb reboot

Doze Feature

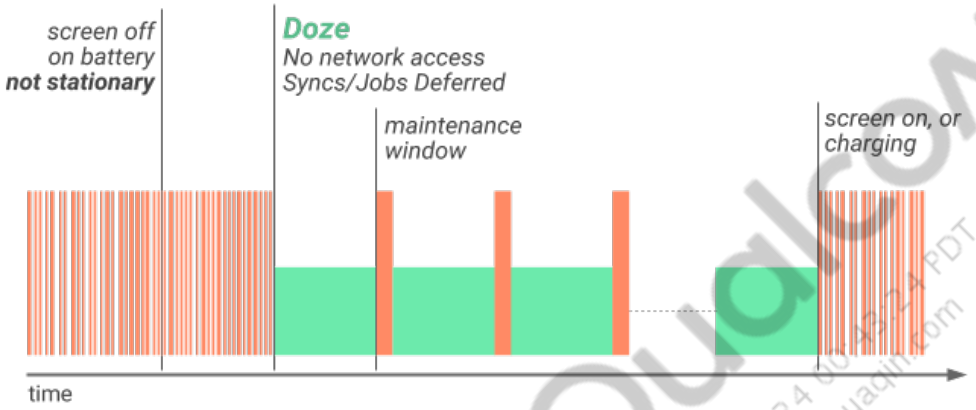


Doze Feature Introduction

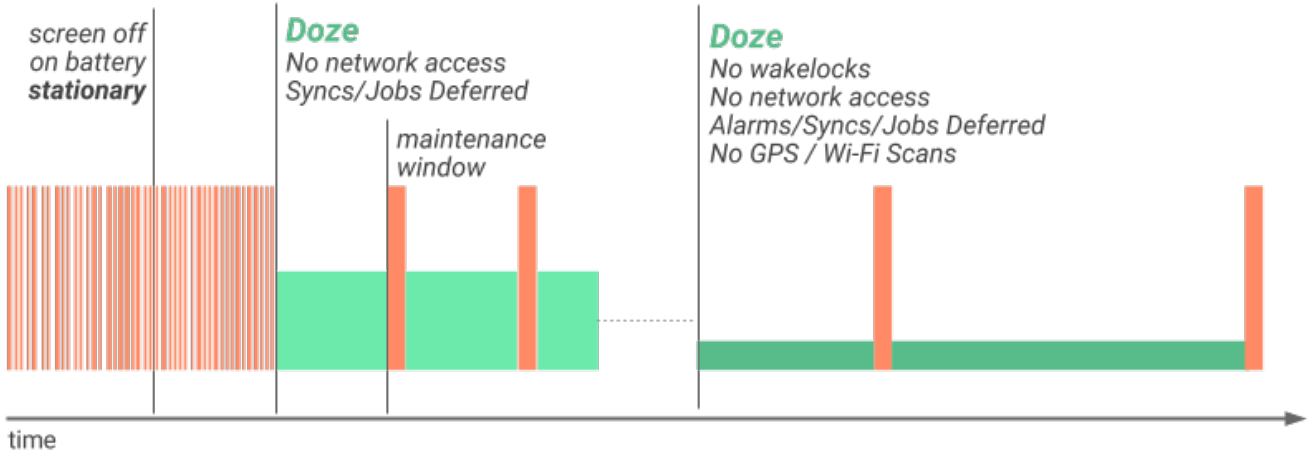
- Doze feature prevents the battery from draining
  - When the device is unplugged with the screen Off for about 30 min, it changes to Doze mode.
  - While Doze is active:
    - Network is not accessible
    - Ignore “wakelocks”, when the applications try to keep the device from going to Sleep
    - Background tasks are not allowed
    - Alarm/Sync deferred
- This original Doze feature called as Doze Deep mode was initially introduced in the Android Marshmallow (Android M) release.
- Later Doze Light mode feature has been introduced in Android Nougat (Android N) release.

Doze Light and Doze Deep Modes

- Doze Light mode starts in Android N after screen Off.



- Doze Deep mode starts after the device remains stationary for long duration (determined by algorithm) after Doze Light.



More Aggressive Doze for Android N Release

- Doze Light mode (new in Android N from Google)
- Doze Light is an extension to Doze feature introduced in Android M release
  - Screen Off and Charging Off indicates LIGHT_STATE_INACTIVE
  - Waits for “n” minutes and checks system activity if any current alarm, wakelock, or network activity is present
  - If there is any pending system activity, the Doze light mode goes into LIGHT_STATE_PRE_IDLE and starts preidle timer with LIGHT_PRE_IDLE_TIMEOUT, and then moves to LIGHT_STATE_IDLE when the timer gets expired; else the system goes to LIGHT_STATE_IDLE
  - Wakelocks and alarms are available

More Aggressive Doze for Android N release (cont.)

- Doze Deep mode (old Android Doze feature from Google)
  - Waits for “n” minutes and starts Motion Detection algorithm to check if the device is idle for sufficient amount of time.
  - If the device is idle and stationary, Doze Deep mode goes to STATE_IDLE; else, the device goes to STATE_INACTIVE and keeps cycling STATE_IDLE_PENDING, STATE_SENSING and STATE_LOCATING states depending on device motion or stationary.
  - No wakelocks are allowed.
  - Alarms are deferred to next maintenance window.
  - No GPS/Wi-Fi scans are allowed.
- Both these modes work independent of each other except when Deep mode is in STATE_IDLE or STATE_IDLE_MAINTENANCE then Light mode will be in LIGHT_STATE_OVERRIDE.

Doze Mode States

- Doze Light mode states

State	Description
LIGHT_STATE_ACTIVE	Device is currently active
LIGHT_STATE_INACTIVE	Screen Off and charging Off, and waits to move into first Light Idle
LIGHT_STATE_PRE_IDLE	Device is about to go Idle for the first time, waits for the current work to complete
LIGHT_STATE_IDLE	Device in Light Idle state, tries to stay asleep as much as possible
LIGHT_STATE_IDLE_MAINTENANCE	Device is in Light Idle state but remains in regular Maintenance mode
LIGHT_STATE_OVERRIDE	Light Idle state is overridden; the device moves to Deep Doze state
LIGHT_STATE_WAITING_FOR_NETWORK	Device is in Light Idle state and wants to go into Idle Maintenance, but its waiting for network connectivity before doing so

- Doze Deep mode states

State	Description
STATE_ACTIVE	Device is currently active
STATE_INACTIVE	Device is inactive (screen Off, no motion) and waits for Idle
STATE_IDLE_PENDING	Device is past the initial inactive period, and waits for the next Idle period
STATE_SENSING	Device is currently sensing motion
STATE_LOCATING	Device is currently finding location (and may still be sensing)
STATE_IDLE	Device is in the Idle state, tries to stay asleep as much as possible
STATE_IDLE_MAINTENANCE	Device is in the Idle state, but temporarily out of idle to do regular maintenance

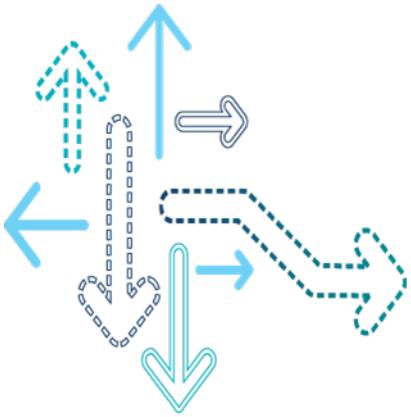
NSRM Coexistence with Doze

Feature	Doze disabled	Doze enabled
NSRM disabled	No power management	Doze power management
NSRM enabled	NSRM power management	Configure NSRM gate opened when Doze Light is in LIGHT_STATE_IDLE or LIGHT_STATE_IDLE_MAINTENANCE and Doze Deep is in STATE_IDLE or STATE_IDLE_MAINTENANCE => Vendor Trigger State to 1 (Enabled) NSRM works as usual in all other Doze states => Vendor Trigger State to 0 (Disabled)

Notes:

- Doze Light LIGHT_STATE_OVERRIDE is analogous to Doze Deep STATE_IDLE and STATE_IDLE_MAINTENANCE states. That is, the override for light state is achieved only when Doze Deep is in idle or idle maintenance. NSRM follows rules for Doze Deep states in this case.
- Doze Light LIGHT_STATE_WAITING_FOR_NETWORK is same as LIGHT_STATE_IDLE. So, NSRM follows the same rules as for the LIGHT_STATE_IDLE.

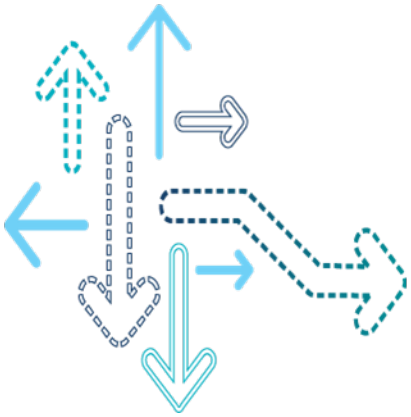
Dataservices Process



New Host Process .dataservices

- Until Android M, devices had independent processes, such as CNEService, DPMService and QtiTetherService.
- From Android N, ".dataservices" is the host process of these three services:
  - com.quicinc.cne.CNEService installed by CNEService.apk
  - com.qti.dpmserviceapp installed by dpmserviceapp.apk
  - com.qualcomm.qti.tetherservice installed by QtiTetherService.apk
- Pros
  - A single common process reduces memory footprint of these data services, this way only one process exists instead of three.
  - Less memory overhead as JVM is shared amongst services.
- Cons
  - If someone manually kills .dataservices process, then all three feature functionalities are impacted momentarily. However, the software is designed to recover from restart of .dataservices process.

DPM Logging



Basic Logging for DPM Issues

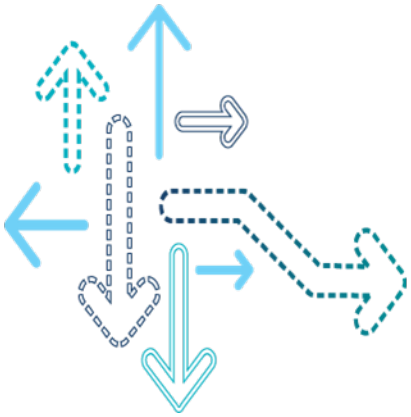
- adb logcat, kernel and tcpdump commands

```
adb logcat -b main -b radio -b system -b events -v threadtime | tee log.log
adb shell cat /proc/kmsg | tee kmsg.log
adb shell tcpdump -nvi rmnet0 -s 0 -w /sdcard/tcpdump.pcap
```

- DPM log messages should be captured in **QXDM Professional log by enabling DPM [10400 ~ 10414] – All level of messages.**

persist.dpm.loglevel	dpmd logs	Java logs
Not set	QXDM (E+W+I)	ADB (E+W+I)
3974	QXDM (E+W+I+D)	Complete logs
7825	QXDM (E+W+I+D+V)	Complete logs

Reference Logs



NSRM Gate State Change and Socket Synchronization

03-20 03:51:01.216 283 283 V DPM : |NSRM:GATESM| void DpmNsrmGateState::HdmiStateInd(NsrmHDMIStateEnum_t):592 Ind:HDMI is connected.

03-20 03:51:01.216 283 283 V DPM : |NSRM:GATESM| void DpmNsrmGateState::TransitionState():877 Current State: Gate is closed., Event Mask: 0x30a Event Enable Mask: 0xd00

03-20 03:51:01.216 283 283 V DPM : |NSRM:GATESM| New State: Gate is open.

03-20 03:51:01.216 283 283 V DPM : |NSRM| Release 4 number of uids.

03-20 03:51:01.216 283 283 V DPM : |NSRM| Release 1 number of sockets belonging to uid 10006

03-20 03:51:01.216 283 283 V DPM : |NSRM| Release socket 3 belonging to uid 10006

03-20 03:51:01.217 283 283 V DPM : |NSRM| Release 2 number of sockets belonging to uid 10008

03-20 03:51:01.217 283 283 D DPM : |NSRM:GATESM| Topen timer of 10 seconds has started.

03-20 03:51:01.217 283 283 V DPM : |NSRM:GATESM| void DpmNsrmGateState::TOpenTimerInd(NsrmTopenTimerStateEnum_t):782 Ind:Topen timer is started.

03-20 03:51:11.219 283 283 V DPM : |NSRM:GATESM| void DpmNsrmGateState::TOpenTimerInd(NsrmTopenTimerStateEnum_t):782 Ind:Topen timer is expired.

03-20 03:51:42.671 283 283 V DPM : |NSRM:GATESM| void DpmNsrmGateState::HdmiStateInd(NsrmHDMIStateEnum_t):592 Ind:HDMI is disconnected.

03-20 03:51:42.671 283 283 V DPM : |NSRM:GATESM| New State: Gate is closed.

03-20 03:51:46.746 283 283 V DPM : |NSRM| NsrmSockReleasePermitType_t DpmNsrmSOI::Synchronize(int, NsrmSocketClassType, unsigned int*, int):401 socketType 0

03-20 03:51:52.181 283 283 D DPM : |NSRM| com.android.browser appname wasnt found.

03-20 03:51:52.181 283 283 V DPM : |NSRM| No appname found. Mode: exclusion.

03-20 03:51:52.181 283 283 V DPM : |NSRM| Synchronize: Write socket is present in queue

03-20 03:51:52.182 283 283 V DPM : |NSRM| Total number of unique uids queued: 2

03-20 03:51:52.182 283 283 D DPM : |NSRM:GATESM| void DpmNsrmGateState::startTSyncTimer(NsrmTimerClassType, int):261

03-20 03:51:52.182 283 283 D DPM : |NSRM:GATESM| getExpectedNToExpiry: lastDataActivityTime = 27 seconds, ntoValue = 300 seconds NTO_TIME_MARGIN = 10

03-20 03:51:52.182 283 283 V DPM : |NSRM:GATESM| Started Twsync timer id 5, actual delay 247591 ms

03-20 03:52:15.194 283 283 D DPM : |NSRM| com.android.vending appname wasnt found.

03-20 03:52:15.194 283 283 D DPM : |NSRM| com.android.vending appname wasnt found.

03-20 03:52:15.194 283 283 V DPM : |NSRM| DpmRetType DpmNsrmApplication::AddSocket(unsigned int):69

Go Dormancy and Out of Dormancy

01-02 08:47:40.055 726 726 I DPM : |FDMGR| DpmFdIdleTimeTracker::idleTimerExpiryHandler - idleStatus = 1
01-02 08:47:40.055 726 726 V DPM : |FDMGR| DpmFdMgr::ifIdleStatusChgEvtHdlr idleStatus = 1
01-02 08:47:40.055 726 726 V DPM : |FDMGR| DpmFdMgr::handleIfIdleStatusChg
01-02 08:47:40.055 726 726 D DPM : |FDMGR| All active ifaces are idle..
01-02 08:47:40.056 726 726 D DPM : |TCM| DpmTcm::dsmEventHandler DPM_DSM_CLOSE_TCP_IDLE_CONNECTIONS
01-02 08:47:40.056 726 726 V DPM : |TCM| DpmTcm::sendCloseTcpIdleConnInd close idle connection
01-02 08:47:40.056 726 726 D DPM : |TCM| DpmTcmServer::closeTcpIdleConnection number of registered connections: 7
01-02 08:47:41.057 726 726 V DPM : |FDMGR| DpmFdMgr::goDormantCallback
01-02 08:47:41.059 726 726 D DPM : |COMMON:QMI| DpmWdsTracker::goDormant: iface: rmnet_data0 go dormant
01-02 08:47:41.291 726 726 D DPM : |COMMON:QMI| DpmWdsTracker::getDormancy: iface rmnet_data0 dormancy status 0
01-02 08:47:41.291 726 726 V DPM : |CTMGR| DpmIfConnTracker::qmiWdsEventHandler event: 0
01-02 08:47:41.291 726 726 V DPM : |CTMGR| DpmIfConnTracker::qmiWdsEventHandler iface: rmnet_data0 status: 0
01-02 08:47:41.291 726 726 V DPM : |NSRM:TRG| DpmNsrmBackgroundEvtHdlr::DormancyChgEvtHandler iface: rmnet_data0 status: 0
01-02 08:47:41.291 726 726 V DPM : |NSRM:GATESM| void DpmNsrmState::WwanConnectivityInd(NsrmWwanConnectivityStateEnum_e):164 Ind:WWAN is idle. Data activity is dormant.
01-02 08:47:41.291 726 726 V DPM : |NSRM:GATESM| RRC State(valid only if wwan connected): 0 (0:not connected; 1:connected)

01-02 08:48:27.697 726 726 I DPM : |FDMGR| DpmFdIdleTimeTracker::idleTimerExpiryHandler - idleStatus = 0
01-02 08:48:27.697 726 726 V DPM : |FDMGR| DpmFdMgr::ifIdleStatusChgEvtHdlr idleStatus = 0
01-02 08:48:27.697 726 726 D DPM : |FDMGR| iface: rmnet_data0 is not idle
01-02 08:48:27.712 726 726 V DPM : |NSRM:TRG| DpmNsrmBackgroundEvtHdlr::DormancyChgEvtHandler iface: rmnet_data0 status: -1
01-02 08:48:27.712 726 726 V DPM : |NSRM:GATESM| void DpmNsrmState::WwanConnectivityInd(NsrmWwanConnectivityStateEnum_e):164 Ind:WWAN is connected. Data activity is not dormant.
01-02 08:48:27.712 726 726 V DPM : |NSRM:GATESM| RRC State(valid only if wwan connected): 1 (0:not connected; 1:connected)
01-02 08:48:27.820 726 726 V DPM : |CTMGR| DpmConnTrackMgr::dsmEventHandler status: 4 iface: rmnet_data0
01-02 08:48:27.820 726 726 V DPM : |CTMGR| DpmIfConnTracker::stopNlSockEvents: deregistering nlsock for ifaceName rmnet_data0

References

Title	Number
Qualcomm Technologies, Inc.	
Data Power Manager API Interface Specification	80-NM328-61
Resources	
https://developer.android.com/preview/behavior-changes.html	–

Acronym or term	Definition
CT	Connection Tracking
DPM	Data Power Management
DNSRM	Dynamically Enable NSRM 2.0 Per Application
ELIST	Exclusion List
FD	Fast Dormancy
ILIST	Inclusion List
NSRM	Network Socket Request Manager
TCM	TCP Connection Manager

Questions?

https://createpoint.qti.qualcomm.com