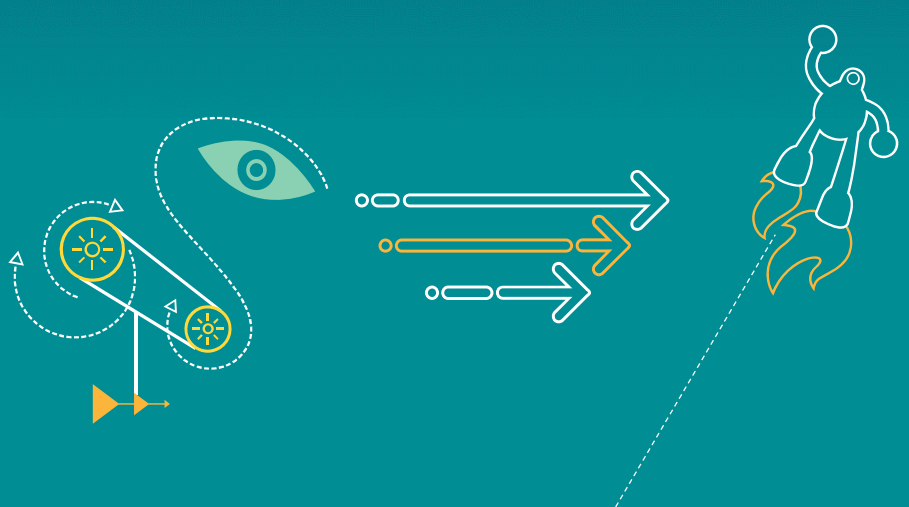
Data Power Manager (DPM) Overview

Qualcomm Technologies, Inc.



80-NU566-1 C

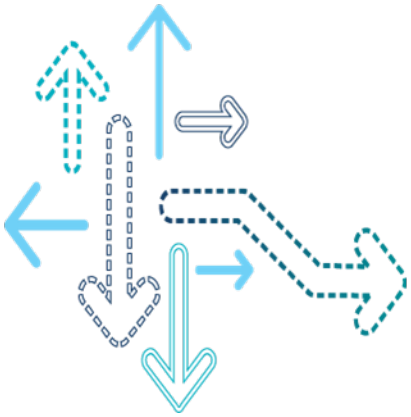
Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	Jan 2015	Initial release.
B	Oct 2015	Added Changes on M Release section.
C	Jan 2016	Added information for Dynamically Enable NSRM 2.0

Contents

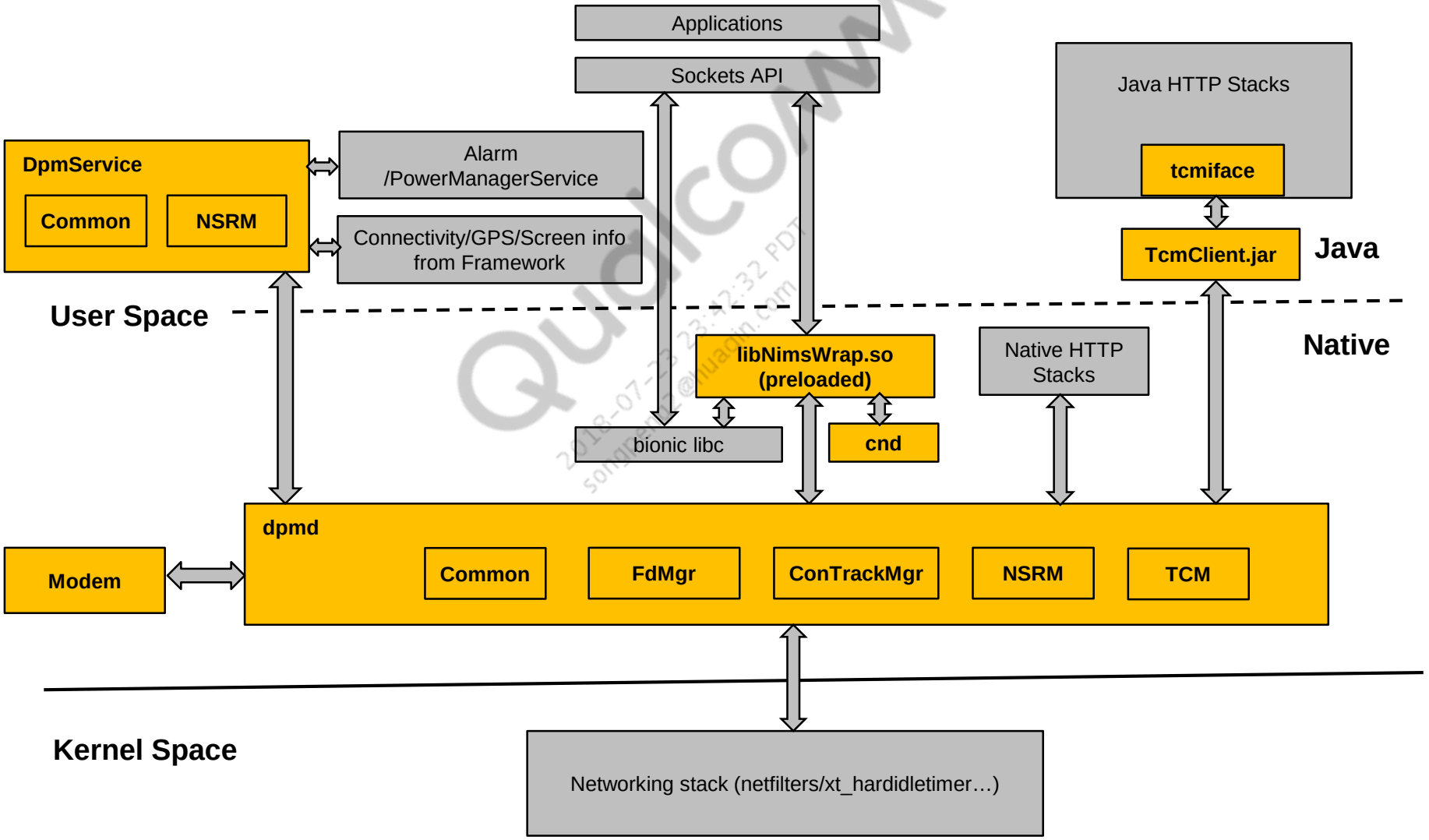
Introduction



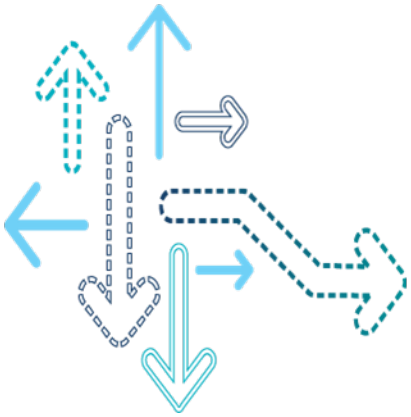
Data Power Manager

- 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
  - persist.dpm.feature (32-bit mask)
  - FD enable or disable – 0x00000001 (first bit 2 – LSB)
  - CT enable or disable – 0x00000002 (second bit)
  - NSRM enable or disable – 0x00000004 (third bit)
  - TCM enable or disable – 0x00000008 (fourth bit)

High-level Architecture



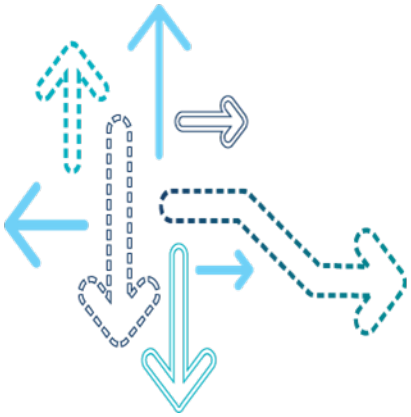
Changes on M Release



DPM Changes on M release

- All DPM features are enabled by default on M
  - Must create and configure persist.dpm.feature to disable specific features
- NSRM is compatible with doze, which is a Google feature to save power by restricting background applications
- Enable or disable TCM at runtime
- NSRM 2.0
  - NSRM 1.0 is enabled by default, the same as other DPM features
  - NSRM 2.0 is enabled by default starting with MSM8953 LA 1.0

Fast Dormancy



Fast Dormancy Problem and Solution

- Problem
  - 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 secs
      - OFF: y secs
    - Tethering state (RNDIS SoftAp only)
      - ON (a cellular interface): z secs, irrespective of the screen state
  - When tethering is active, the tethering timer is used, irrespective of the screen state.

Fast Dormancy Config File

- Simple 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 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 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 next slide for the bit selection of each network type)

Network Types for Selected Bitmask Value

- #Fast dormancy and TCM can be 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

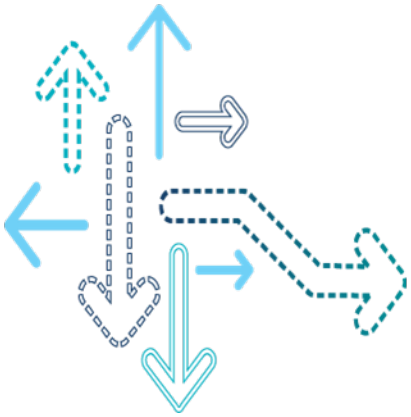
Recommended Configuration

- The default recommended configuration file is present in one for the following:
  - /system/etc/dpm/fdmgr/fd.conf (targets *not* supporting TCM)
  - /system/etc/dpm/dpm.conf (targets supporting TCM)
- OEMs can push their own configuration file to one for the following:
  - /data/dpm/fdmgr/fd.conf (targets *not* supporting TCM)
  - /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. If for some reason the file in that location becomes 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. If you do not want to disturb 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 config, do one of the following:
       - adb push fd.conf /data/dpm/fdmgr/fd.conf
       - abd 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. If you do not want to disturb 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

- Problem
  - 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 can filter spurious download packets and not wake up the apps processor unnecessarily
    - On spurious packet reception, the modem can aggressively go dormant
  - This saves power in networks with spurious packets
    - Less apps processor wake time
    - More modem RRC dormant time

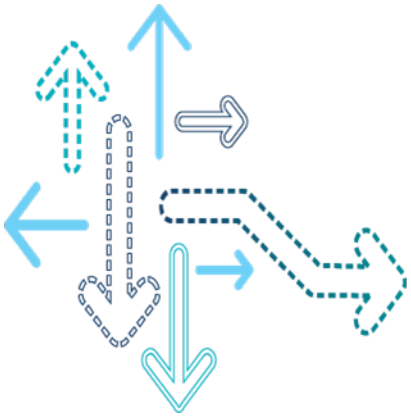
How to Enable or Disable CT

- 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. If you do not want to 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 next slide)
- To disable CT:
  1. adb root
  2. adb wait-for-device
  3. adb setprop persist.dpm.feature 0
     - This disables all the DPM features. If you do not want to 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 next slide)

How to Enable or Disable CT (cont.)

- Note that the CT feature works in conjunction 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 an RAB for data transfer and there no activity for 6 sec (configurable in EFS with TIMER_VALUE_1 below).
- Enable or disable modem FD and filtering, and configure FD timer values
  - To enable or disable the FD feature, push the 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 the 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.

Network Socket Request Manager

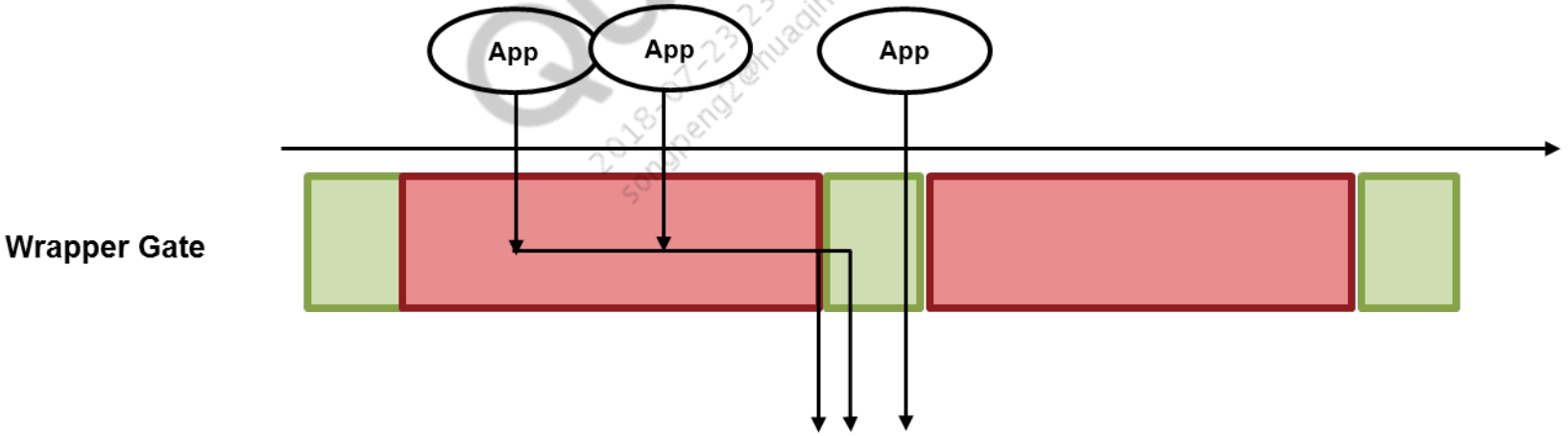


NSRM

- Problem
  - Applications such as Facebook, Twitter, and so on, periodically pull data from the network and the requests are asynchronous with regard to time. This results in more RRC connections, that is, 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, and so on)
    - Reduces network signaling
    - Saves device power
    - Has no impact on user experience
  - The algorithm is based on the concept of the NSRM gate.

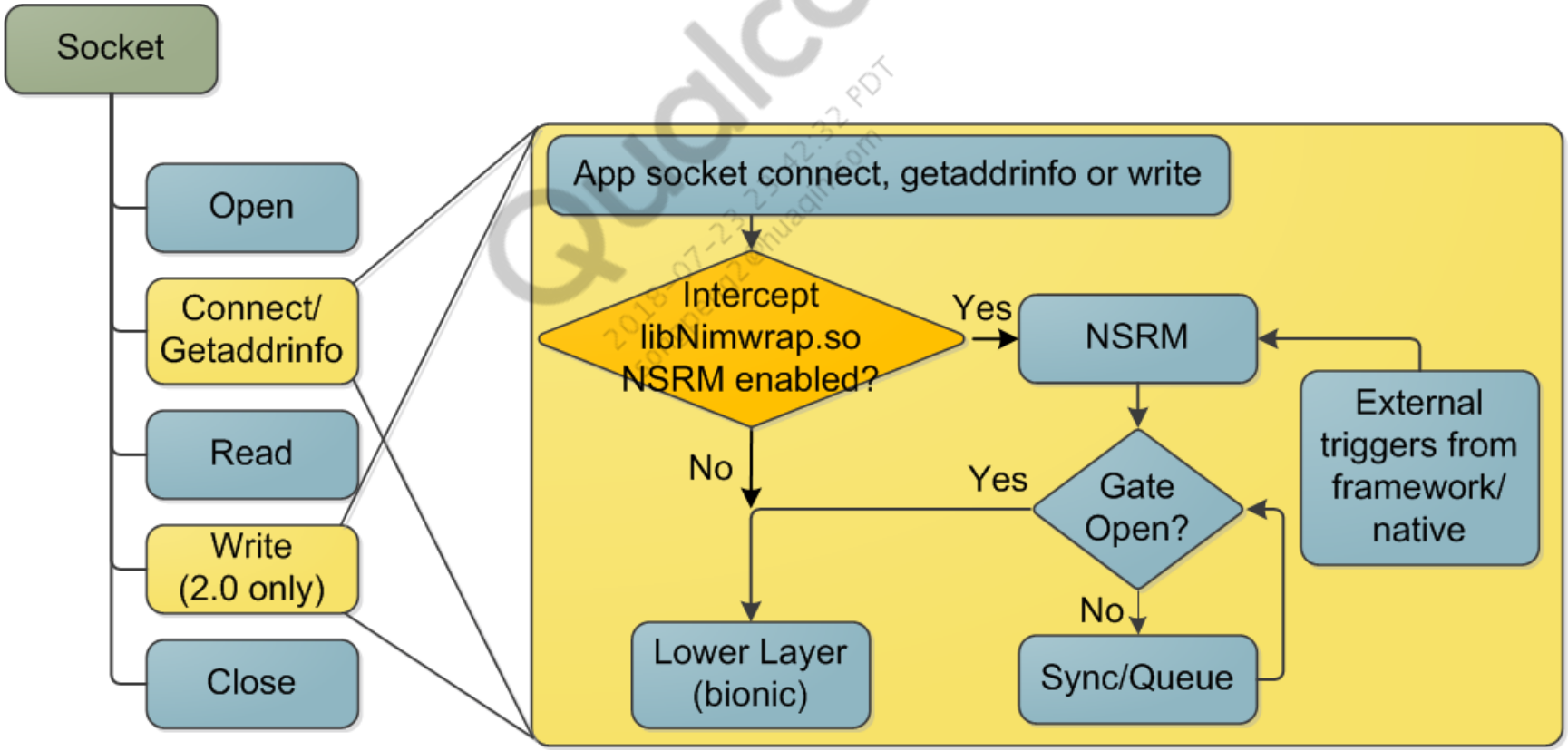
NSRM Gate

- 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

- 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. First location is /data/dpm/nsrm/
  2. If it does not find the file at the first location or the parsing of the file failed, NSRM looks for the file at /system/etc/dpm/nsrm
  3. If it does not find any file at this second location or the parsing fails, NSRM falls back to default hardcoded values for the configuration parameters
- In the builds released by Qualcomm, the recommended configuration file is present in /system/etc/dpm/nsrm; and /data/dpm/nsrm does not have any configuration file
- OEM applications can 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 2.0 Configuration Example

```
<?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>
```


NSRM 1.0 Configuration Example

- For the older NSRM version 1.0, the configuration file format is different
  - NSRM 1.0 does not synchronize the write calls, so there is no Tnto value or Twsync timer
  - Tssync timer is called Tsync timer in NSRM 1.0, and it starts whenever the gate state changes to Close

```
<?xml version="1.0" encoding="UTF-8"?>
<NsrmPolicy>
  <Nsrm>
    <!--The version of the NSRM configuration file format-->
    <Version>1.0</Version>
    <!--Length of time in seconds to keep gate open each time it is (Topen)-->
    <GateOpenTime>10</GateOpenTime>
    <!--Maximum time in seconds to wait before forcing gate open for connect or getaddrinfo (Tssync)-->
    <GateSyncTime>1200</GateSyncTime>
    <!--Mode to indicate how processes which share the same UID should be handled-->
    <SharedUIDMode>Conservative</SharedUIDMode>
    <!--List of applications which to apply Nsrm to-->
    <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>
  </Nsrm>
</NsrmPolicy>
```


Tssync and Twsync 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.
- The timer values are configurable.

Topen Timer

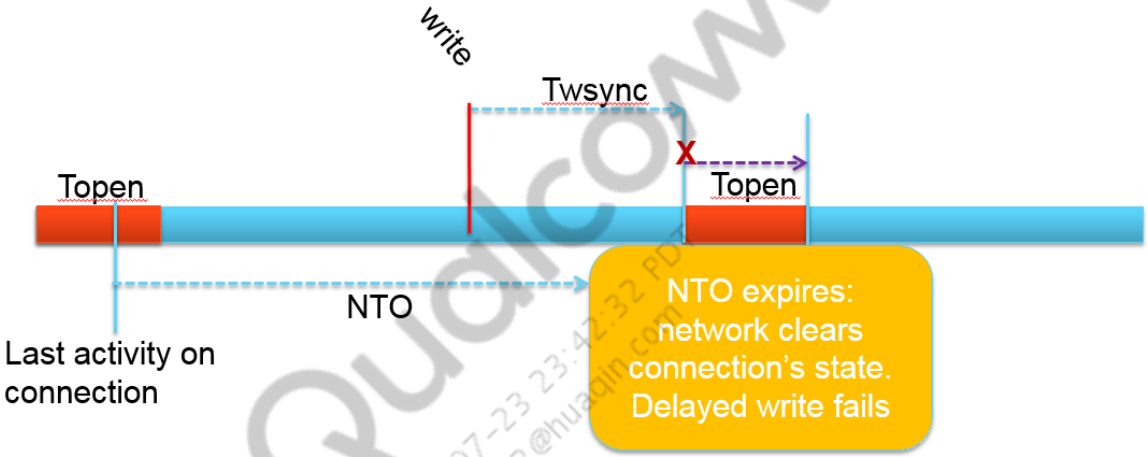
- This timer starts when the gate state changes to Open, and keeps the gate state Open until the timer expires. After the timer expires, if all the other conditions are met, the gate state changes to Close.
- The timer value is configurable.

Tnto

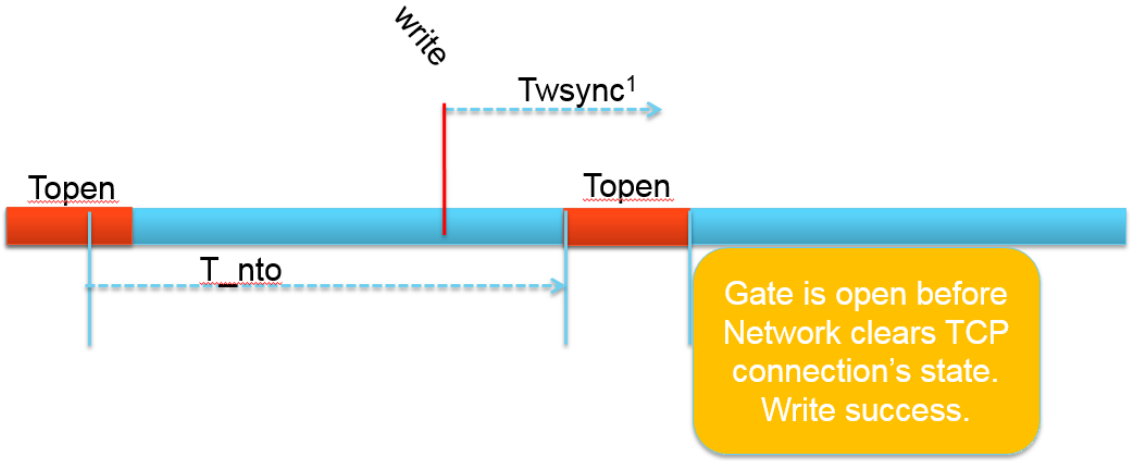
- Tnto ensures that NSRM does not synchronize write traffic beyond the inactivity time used by the NAT and Firewalls.
  - Stateful middleboxes in the network, such as NAT or Firewalls, must maintain the state of each connection.
  - The state of the NAT or Firewall is erased after a certain amount of inactivity 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.

Tnto Example

- Twsync delaying beyond network inactivity



- Opening the gate to avoid exceeding NTO



Tnto Value Configuration

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

Tnto Value Configuration Example

- 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 in 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 can be informed with a short inactivity timeout (5 min). Alternatively, the delay of write() can be turned off in those countries by setting a 0 value, as shown in the example on the next slide.

Tnto Value Configuration Example (cont.)

- How to set the relevant portion of NSRM configuration for this example:

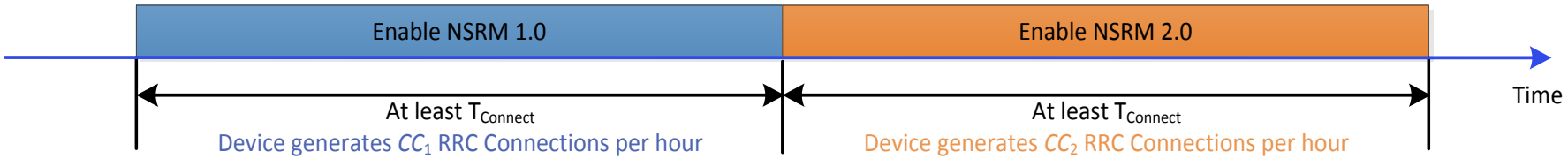
```
<MCC_MNC value="123987">
          <port value="80" NTO="1200"/>
          <port value="413" NTO="300"/>
          <port value="0" NTO="900"/>
</MCC_MNC>
<MCC_MNC value="465753">
          <port value="5228" NTO="2100"/>
          <port value="0" NTO="780"/>
</MCC_MNC>
<MCC_MNC value="Default">
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

Dynamically Enable NSRM 2.0 (DNSRM)

- Problem
  - NSRM 2.0 delays write() calls and sometimes causes a problem.
    - 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 cannot be done to find all the NSRM non-friendly applications.
- Solution
  - Only NSRM 1.0 will be enabled by default, which reduces the gain.
  - Detect dynamically if NSRM is performing well on the UE.
  - Enable NSRM 2.0 dynamically per application.

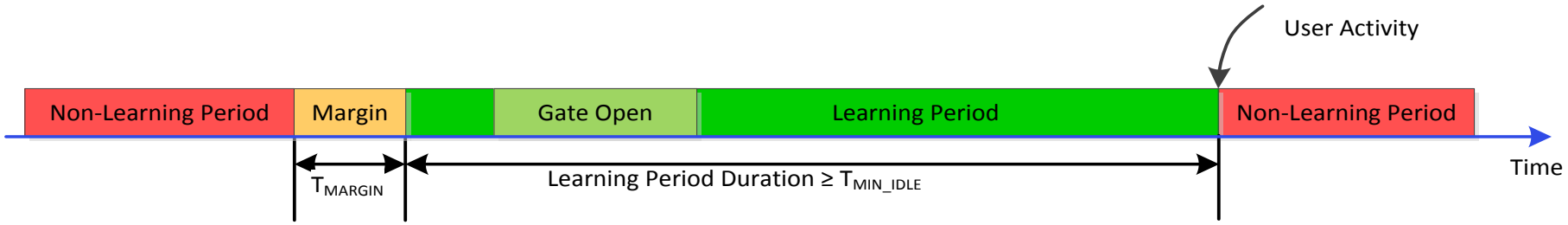
Dynamically Enable NSRM 2.0 – Overview

- 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.
  - It is 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
- TConnect – Minimum duration of the connection count
  - DNSRM counts connections for at least TConnect 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 the 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

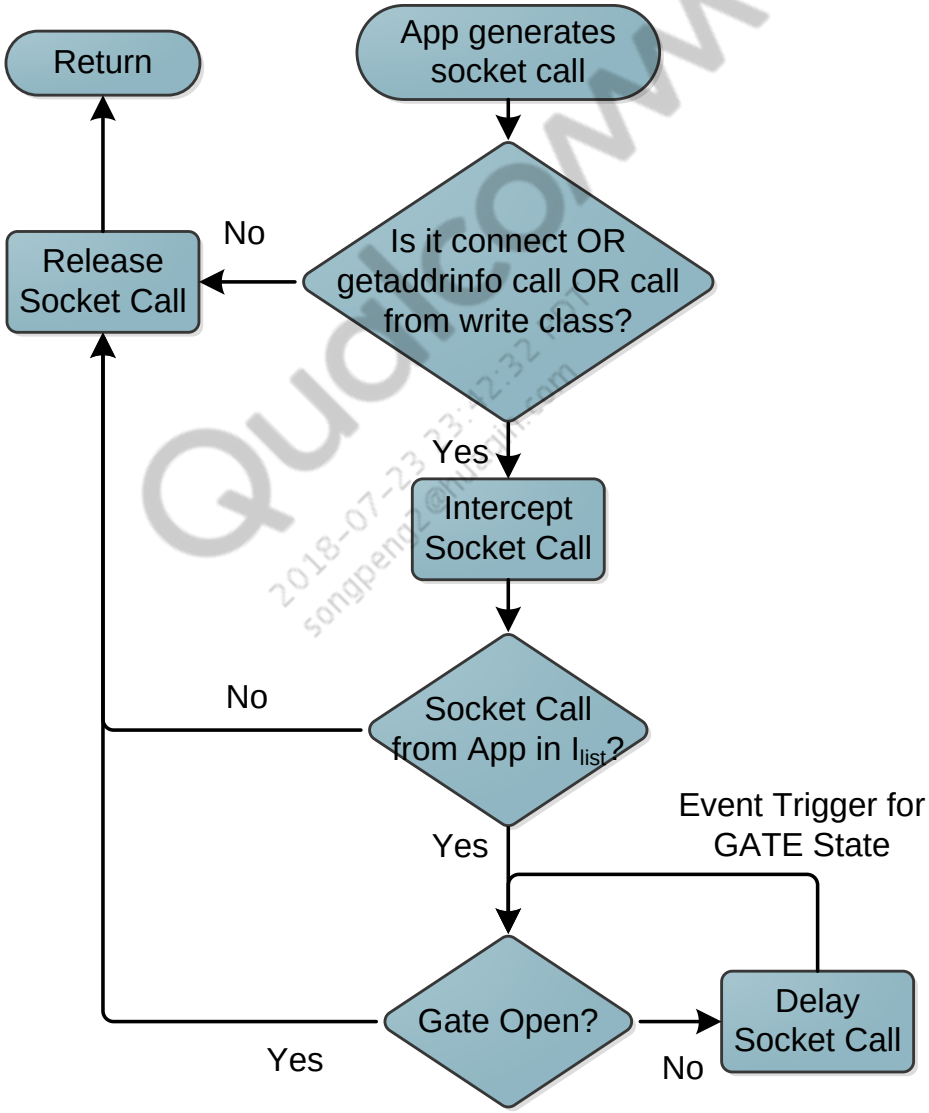
Emergency Alert

- The reception of an emergency alert can trigger opening 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 uniformly 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.

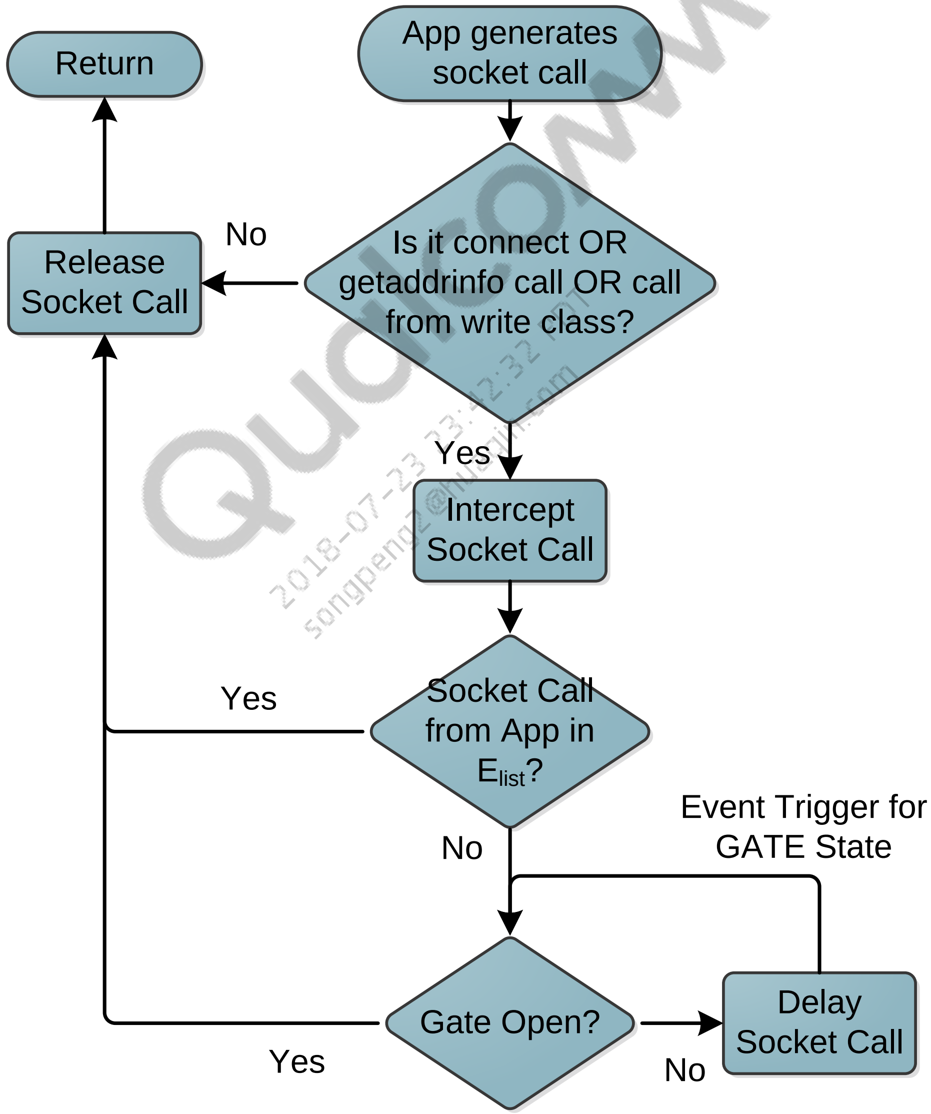
NSRM Modes Of Operation

- NSRM can operate 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 can be conservative or aggressive.
- These settings can be 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 are in the list	**Any** application with the same UID is **not** in the list
Aggressive	**Any** application with the same UID is in the list	**All** applications with the same UID are **not** in the list

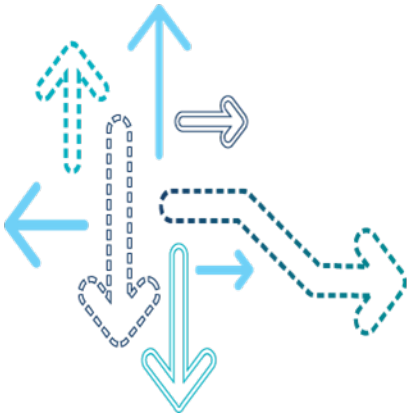
Aggressive vs Conservative (cont.)

- 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. If you do not want to 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.
     - If you want to update the config:
       adb push NsrmConfiguration.xml /data/dpm/nsrm/NsrmConfiguration.xml
     - If you want 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. If you do not want to 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. abd reboot

TCP Connection Manager



TCP Connection Manager

- Problem
  - Popular Android applications such 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

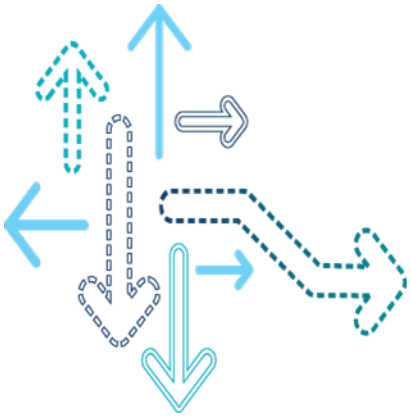
```
#Idle timer value in seconds when SCREEN state is ON
dpm_tcm_screen_on_idle_timer_value:5
#Idle timer value in seconds when SCREEN state is OFF
dpm_tcm_screen_off_idle_timer_value:1
#TCM can be configured for a network type; it's a bit mask
#Default configuration 1111111111111110  (Enabled everything)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

- The default recommended configuration file for DPM is present in /system/etc/dpm/dpm.conf
- OEMs can push their own 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 becomes corrupted and parsing fails, the parser uses the software hardcoded recommended values from above.

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. If you do not want to 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.
     - If you want to update the config:
       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. If you do not want to 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. abd reboot

DPM Logging



Basic Logging for issues

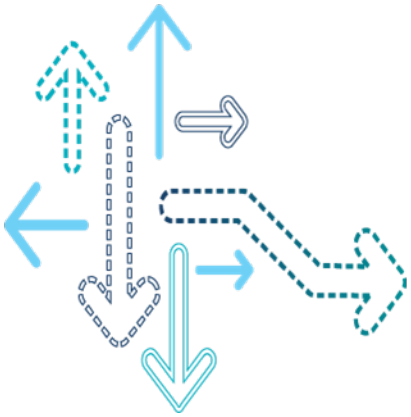
- Commands
  - adb logcat -b main -b radio -b system -b events -v threadtime | tee log.log
  - adb shell cat /proc/kmsg | tee kmsg.log
  - adb shell tcpdump -nvi rmnet0 -s 0 -w /sdcard/tcpdump.pcap
  - QXDM Professional™ log with SSIDs DPM, legacy, QMI, and Data Service messages enabled
- For any issue, start with these four basic types of logs

Logging with QXDM Pro vs ADB

Sl.No.	libdpmlog.so present in /data/dpm/	Persist.dpm.loglevel	dpmd logs	java logs
1	No	Not set	QXDM(E+W+I)	ADB(E+W+I)
2	No	3974	1 + DEBUG	Complete logs
3	No	7825	2 + VERBOSE	Complete logs
4	Yes	Not Set	ADB(E+W+I)	ADB(E+W+I)
5	Yes	3974	4 + DEBUG	Complete logs
6	Yes	7825	5 + VERBOSE	Complete logs

- Complete the logging via adb
  1. adb root
  2. adb remount
     - adb push libdpmlog.so /system/vendor/lib (32-bit libdpmlog.so built for 32-bit release)
     - adb push libdpmlog.so /system/vendor/lib64 (64-bit libdpmlog.so built for 64-bit release)
     - adb push libdpmlog.so /data/dpm/ (Software release without CR 752294 for SEAndroid support)
  3. adb shell setprop persist.dpm.loglevel 7825
- libdpmlog.so can be requested by case owners

References



References

Documents	
Qualcomm Technologies, Inc.	
Data Power Manager API Interface Specification	80-NM328-61

Acronyms	
Term	**Definition**
CT	Connection tracking
DPM	Data Power Manager
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