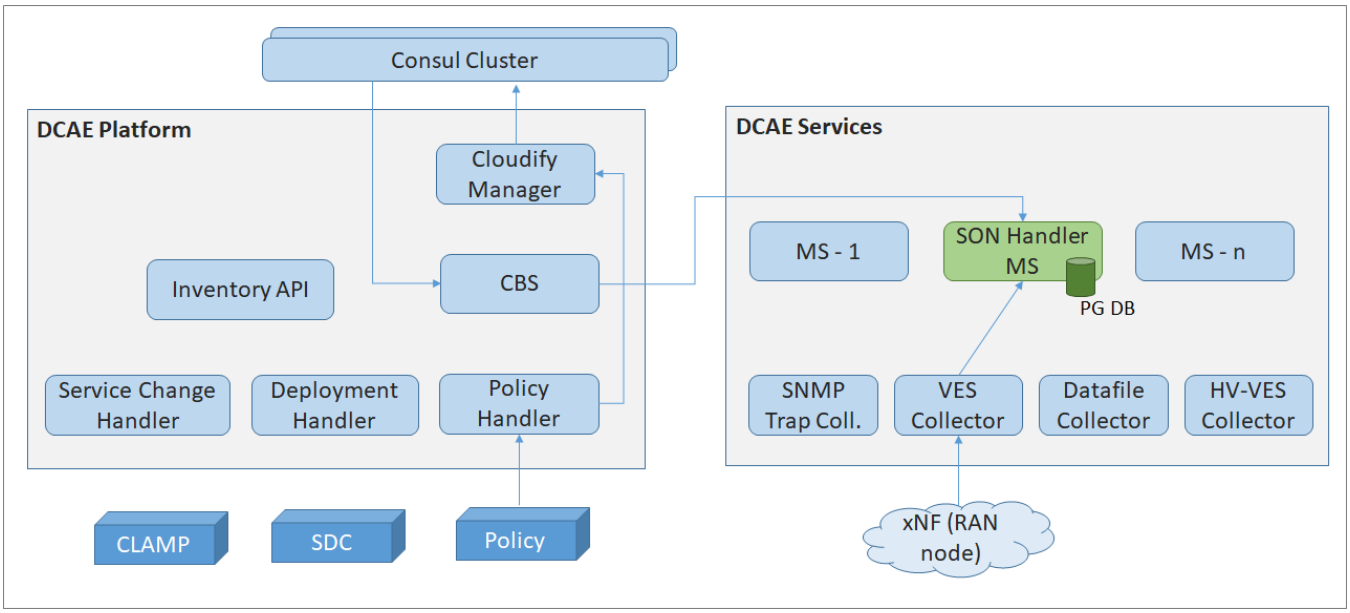
SON-Handler MS (DCAE) Impacts

References

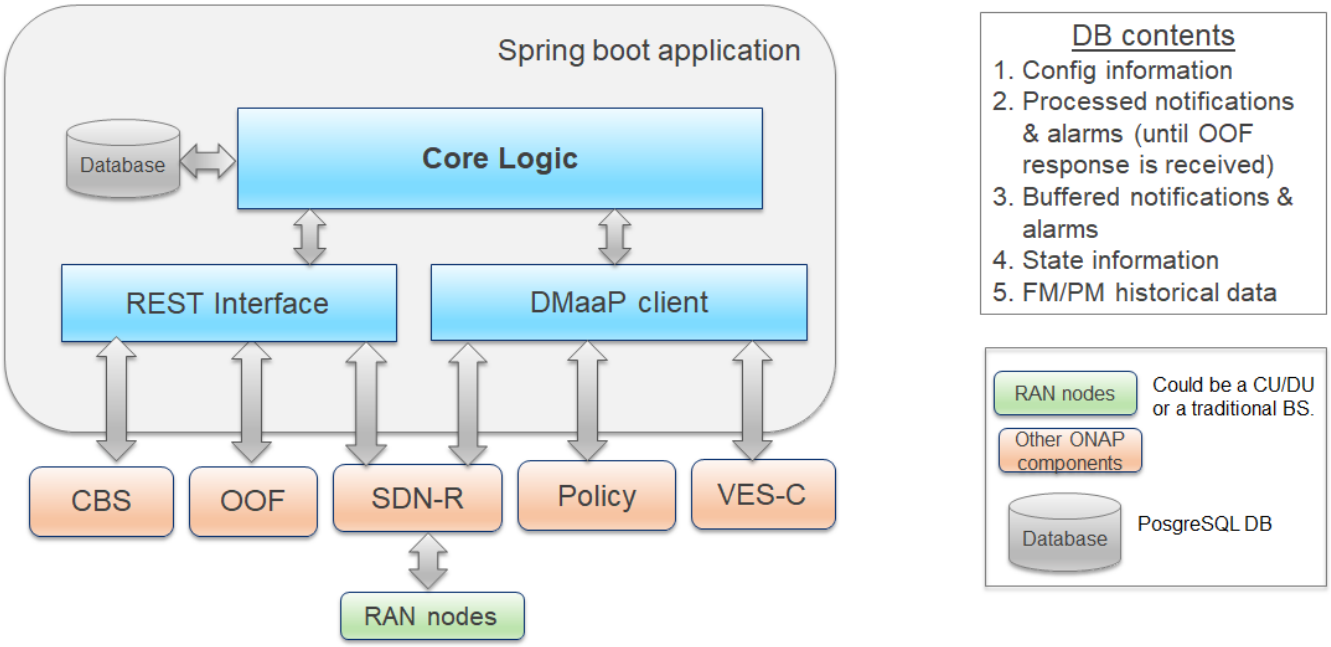
Use case overview and impacted components are described in the 5G OOF PCI optimization main wiki page.

1. Architecture

The architecture below depicts the SON-Handler MS as a part of DCAE. Only the relevant interactions and components are shown.



The internal architecture of SON-Handler MS is shown below.



1.1. Core Logic

This is composed of a Main Thread and one or more Child Threads. The Main Thread is responsible for spawning and terminating the Child Threads, this is described in the sections below.

1.2. Database

This is a PostgreSQL DB, and is intended to persist information such as the following:

- PCI-Handler MS Config information (e.g., thresholds, timer values, OOF algorithm name, etc.)
- Pre-processing results and other related information (e.g., neighbor list)
- Buffered notifications (i.e., notifications not yet processed at all)
- State information
- Association between pnf-name and CellId
- PM/FM data

1.3. DMaaP Client

This is responsible for registering with the DMaaP client for the DMaaP notifications from SDN-R and VES-Collector, and to Policy.

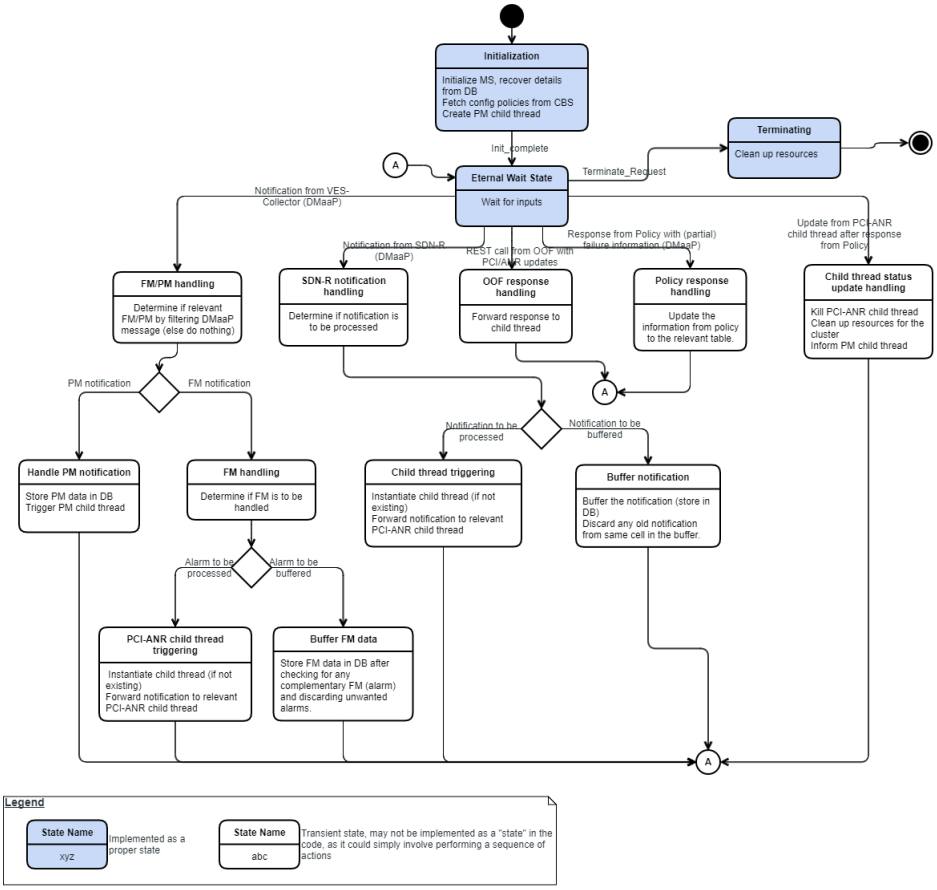
2. Core Logic components and Pre-processing algorithm

Note: The pre-processing logic may not be 100% fool-proof as well as the most efficient one. An attempt has been made to balance the usefulness for a PoC versus the complexity of handling all possible scenarios.

2.1. Main Thread

The state transition diagram is illustrated below:

The main actions in the various states are listed below:



2.1.1. Initialization

In this state, the following actions are performed:

- Fetch Config Policy from CBS.
- Load any local configuration.
- Load data persisted in DB.
- Trigger DMaaP thread for registration of DMaaP topics (SDN-R, VES-Collector and Policy interfaces).

2.1.2. Eternal Wait State

The main thread always comes back to this state (except when terminated). In this state, the inputs and associated actions are summarized below:

2.1.2.1. DMaaP message from VES-Collector received (PM/FM data)

- FM data received: Determine if alarm to be processed, if yes, trigger appropriate PCI-ANR child thread, else, buffer the FM data.
- PM data received: Trigger PM child thread

2.1.2.2. SDN-R Notification handling

Upon receipt of a neighbor-list-change notification message (DMaaP) from SDN-R, do the following:

1. Fetch the PCI_optimization requests (from DB) for which OOF has been triggered, and corresponding 'cluster' details
2. For each 'cluster' for which OOF has been triggered, check if the Nodeid of the Cell Ci or at least one cell in its NbrList matches with any cell in the 'cluster'
3. If there is a match, then the request has to be buffered, along with the cluster indication (cluster id), else the notification has to be processed. If the notification has to be processed, then instantiate the PCI-ANR child thread (if not already active), and forward the notification.

2.1.2.3. OOF Response Handling

Upon invocation of API by OOF for PCI result pass it to the relevant child thread(s) (by request id <-> thread mapping). Store the OOF results in database along with the timestamp.

2.1.2.4. Policy response handling

- Response from policy is handled by a separate thread.
- When positive acknowledgement is received from the policy the cells in the response are removed from the table and when negative acknowledgement is received the negative_ack count for the cell is increased and updated in the table. When no response is received no changes are made.
- When negative_ack count of a particular cell increases beyond a threshold, it is shifted to another table.

2.1.2.5. Child Thread Status update handling

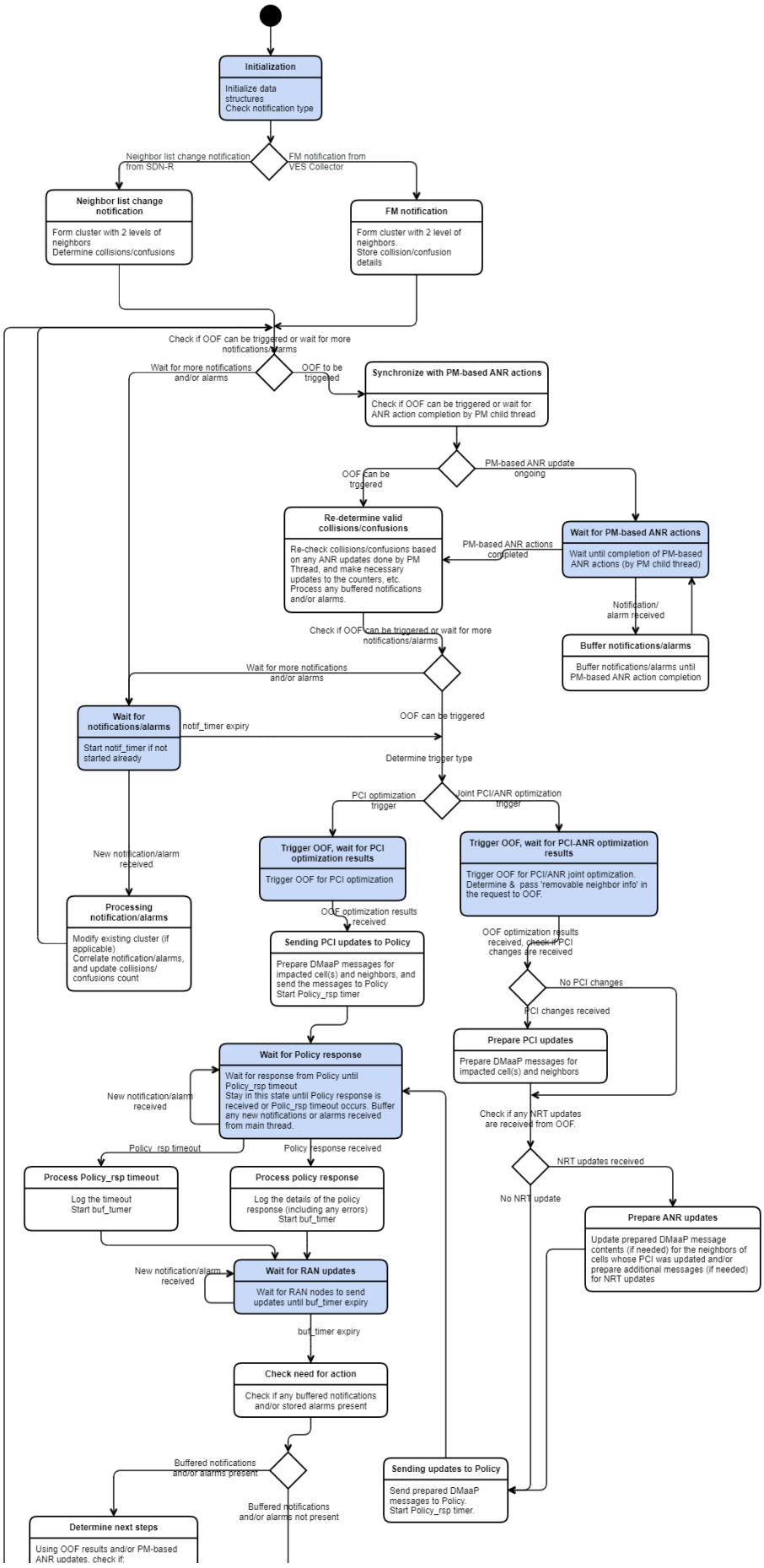
Clean up the resources associated with the child thread (including cluster details), and kill the child thread.

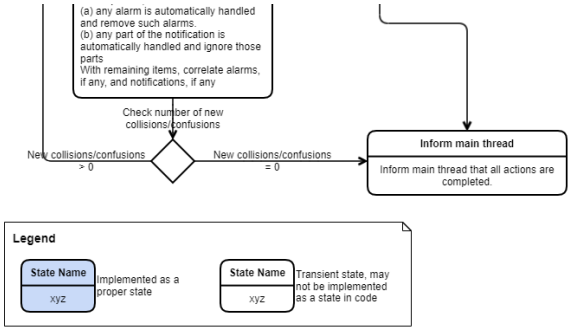
2.1.3. Terminating

Upon receiving a terminate request clean up all resources.

2.2. PCI-ANR Child Thread(s)

These child threads are spawned for handling PCI optimization primarily, though they may also trigger PCI-ANR joint optimization in some cases. The various states and associated actions are described below.


Initialization
Initialize data structures
Check notification type
Neighbor list change notification from SDN-R
FM notification from VES Collector
Neighbor list change notification
Form cluster with 2 levels of neighbors
Determine collisions/confusions
FM notification
Form cluster with 2 level of neighbors.
Store collision/confusion details
Check if OOF can be triggered or wait for more notifications/alarms
Wait for more notifications and/or alarms
OOF to be triggered
Synchronize with PM-based ANR actions
Check if OOF can be triggered or wait for ANR action completion by PM child thread
OOF can be triggered
PM-based ANR update ongoing
Re-determine valid collisions/confusions
Re-check collisions/confusions based on any ANR updates done by PM Thread, and make necessary updates to the counters, etc.
Process any buffered notifications and/or alarms.
PM-based ANR actions completed
Wait for PM-based ANR actions
Wait until completion of PM-based ANR actions (by PM child thread)
Notification/ alarm received
Buffer notifications/alarms
Buffer notifications/alarms until PM-based ANR action completion
Check if OOF can be triggered or wait for more notifications/alarms
Wait for more notifications and/or alarms
Wait for notifications/alarms
Start notif_timer if not started already
notif_timer expiry
OOF can be triggered
Determine trigger type
PCI optimization trigger
Joint PCI/ANR optimization trigger
New notification/alarm received
Trigger OOF, wait for PCI optimization results
Trigger OOF for PCI optimization
Trigger OOF, wait for PCI-ANR optimization results
Trigger OOF for PCI/ANR joint optimization.
Determine & pass 'removable neighbor info' in the request to OOF.
Processing notification/alarms
Modify existing cluster (if applicable)
Correlate notification/alarms, and update collisions/ confusions count
OOF optimization results received
Sending PCI updates to Policy
Prepare DMaaP messages for impacted cell(s) and neighbors, and send the messages to Policy
Start Policy_rsp timer
OOF optimization results received, check if PCI changes are received
No PCI changes
PCI changes received
Prepare PCI updates
Prepare DMaaP messages for impacted cell(s) and neighbors
Wait for Policy response
Wait for response from Policy until Policy_rsp timeout
Stay in this state until Policy response is received or Polic_rsp timeout occurs. Buffer any new notifications or alarms received from main thread.
New notification/alarm received
Check if any NRT updates are received from OOF.
Policy_rsp timeout
Policy response received
NRT updates received
No NRT update
Process Policy_rsp timeout
Log the timeout
Start buf_tumer
Process policy response
Log the details of the policy response (including any errors)
Start buf_timer
Prepare ANR updates
Update prepared DMaaP message contents (if needed) for the neighbors of cells whose PCI was updated and/or prepare additional messages (if needed) for NRT updates
New notification/alarm received
Wait for RAN updates
Wait for RAN nodes to send updates until buf_timer expiry
buf_timer expiry
Check need for action
Check if any buffered notifications and/or stored alarms present
Buffered notifications and/or alarms present
Buffered notifications and/or alarms not present
Sending updates to Policy
Send prepared DMaaP messages to Policy.
Start Policy_rsp timer.
Determine next steps
Using OOF results and/or PM-based ANR updates, check if:



2.2.1 Initialization

In this state, perform initialization, and based on the type of notification, go to either Section 2.2.1.1 or 2.2.1.2 for next actions.

2.2.1.1. Neighbor list change notification received

- Store details of the cell (which sent the neighbor list change notification) and its neighbors in DB.
- For each of the neighbors, fetch the neighbor list (via REST API) from SDN-R Config DB (i.e., fetch neighbor of neighbors of the cell that sent the neighbor list change notification).
- Form a cluster with the cell, its neighbors and neighbors of each of the neighbors.
- Assign a cluster id.
- Determine collision/confusion by calling method *determine_collision_confusion*. Go to step 2.2.1.3.

2.2.1.2. FM notification received

- Store details of the FM in DB.
- Increment the number of collisions and/or confusions depending on the type of alarm received. Go to step 2.2.1.3.

2.2.1.3. Check if OOF can be triggered

Based on number of collisions/confusions and config policy, determine if OOF has to be triggered or more notifications should be awaited (handle also timeout case).

If OOF has to be triggered

then

    if any ANR actions are ongoing by PM-child thread

        Go to "Wait for PM-based ANR actions" state (Section 2.2.2)

    else

        Go to step 2.2.1.4 below.

    fi

else

    Go to "Wait for notifications/alarms" state (Section 2.2.3)

fi

2.2.1.4. Re-determine valid collisions/confusions

This step is to ensure that collisions and confusions are re-computed after PM-based ANR actions are completed. When transitioning from 2.2.1.1 or 2.2.1.2, this action need not be carried out. After re-computation of collisions/confusions, again a check on number of collisions/confusions should be done to determine if OOF has to be triggered or more notifications should be awaited (handle also timeout case).

If OOF has to be triggered

then

    Go to Step 2.2.1.5 below

else

    Go to "Wait for notifications/alarms" state (Section

fi

2.2.1.5. Determine OOF trigger type

A simple logic, for e.g., based on number of times PCI optimization has been triggered during a given time window could be used to determine whether PCI optimization or joint PCI-ANR optimization should be triggered.

If PCI optimization should be triggered

then

    Go to "Trigger OOF, wait for PCI optimization results" state (Section 2.2.4)

else

    Go to "Trigger OOF, wait for PCI-ANR optimization results" state (Section 2.2.5)

fi

2.2.2. Wait for PM-based ANR actions

Wait until PM-based ANR actions are completed by PM child thread. In this state, if any new notifications/alarms are received, then the child thread simply buffers them. Upon knowing that the PM-based ANR actions are completed, go to Step 2.2.1.4.

2.2.3. Wait for notifications/alarms

Wait for more notifications/alarms, and start notification_timer if not started already. Upon reception of a new notification/alarm, go to Step 2.2.3.1 below. Upon notification_timer expiry, go to Step 2.2.1.5.

2.2.3.1. Process notification/alarm

- If a cell that already exists in the cluster has sent a notification (for neighbor-list change), update the cell's neighbors appropriately in the existing cluster. Otherwise, 'attach' the cell appropriately in the existing cluster, and update the cell's neighbors.
- For each of the neighbors, fetch the neighbor list (via REST API) from SDN-R Config DB (i.e., fetch neighbor of neighbors of the cell that sent the neighbor list change notification).
- Modify/extend the cluster appropriately.
- Determine collision/confusion by calling method *determine_collision_confusion*.
- Correlate alarms and neighbor list change notifications (to avoid duplicate counts),
- Go to Step 2.2.1.3.

Note: When transitioning to this state from *Handle buffered notifications* state, more than 1 notification may have to be handled.

2.2.4. Trigger OOF, wait for PCI optimization results

- Send a PCI_opt request to OOF with cells triggering the request along with the trigger type
- Store details of request in DB.
- Wait for OOF optimization result.
- Upon trigger from main thread with OOF PCI optimization result, prepare and send DMaaP messages to Policy: for the cells whose PCI value has changed, as well as to all the neighbors of the cells whose PCI value has changed. The pnf-name corresponding to the cell-ids can be fetched from Config DB of SDN-R (using REST API). (Note: The layout of the message from SDN-R to RAN is available in the SDN-R sub-page).
- Start Policy_rsp timer and wait for Policy response by going to "Wait for Policy Response" state (Section 2.2.6).

2.2.5. Trigger OOF, wait for PCI-ANR optimization results

- Fetch the list of cell pairs whose HO success is less than thresold_poor, but >= threshold_bad (this would have been prepared already by PM child thread and stored in DB).
- Send a Joint_PCI_ANR_opt request to OOF with cells triggering the request along with the trigger type, and list of 'removable neighbors'
- Store details of request in DB.
- Wait for OOF optimization result.
- Upon trigger from main thread with OOF ANR-PCI optimization result

    if response contains PCI updates

    then

        Prepare DMaaP messages to Policy for the cells whose PCI value has changed, as well as to all the neighbors of the cells whose PCI value has changed. The pnf-name corresponding to the cell-ids can be fetched from Config DB of SDN-R (using REST API). (Note: The layout of the message from SDN-R to RAN is available in the SDN-R sub-page)

    fi

    if response contains NRT updates (corresponding to ANR)

        Update/prepare DMaaP messages to Policy for the cells whose neighbors have undergone updates (with indication "HO prohibited"). The pnf-name corresponding to the cell-ids can be fetched from Config DB of SDN-R (using REST API). (Note: The layout of the message from SDN-R to RAN is available in the SDN-R sub-page)

    fi

- Send prepared message(s) to Policy.
- Start Policy_rsp timer and wait for Policy response by going to "Wait for Policy Response" state (Section 2.2.6).

2.2.6. Handling Policy Response

- Response from policy is handled by a separate thread.
- When positive acknowledgement is received from the policy, the cells in the response are removed from the table and when negative acknowledgement is received the negative_ack count for the cell is increased and updated in the table. When no response is received no changes are made.
- When negative_ack count of a particular cell increases beyond a threshold, it is shifted to another table.

2.2.7. Wait for RAN updates

Wait in this state until buf_timer expires. If any notification/alarm is received in this state, simply buffer/store them. Upon expiry of buf_timer, check if there are any buffered notifications.

if there are buffered notifications/alarms

then

- Alarms: Based on OOF results and/or PM-based ANR updates check if any new alarms are automatically inapplicable, and remove them.
- Notifications: Based on OOF results and/or PM-based ANR updates check if any new neighbor list change notifications are automatically inapplicable (partially), and update them.
- Determine collisions/confusions, perform correlation of pending alarms and notifications to remove duplicate entries. Go to Step 2.2.1.3.

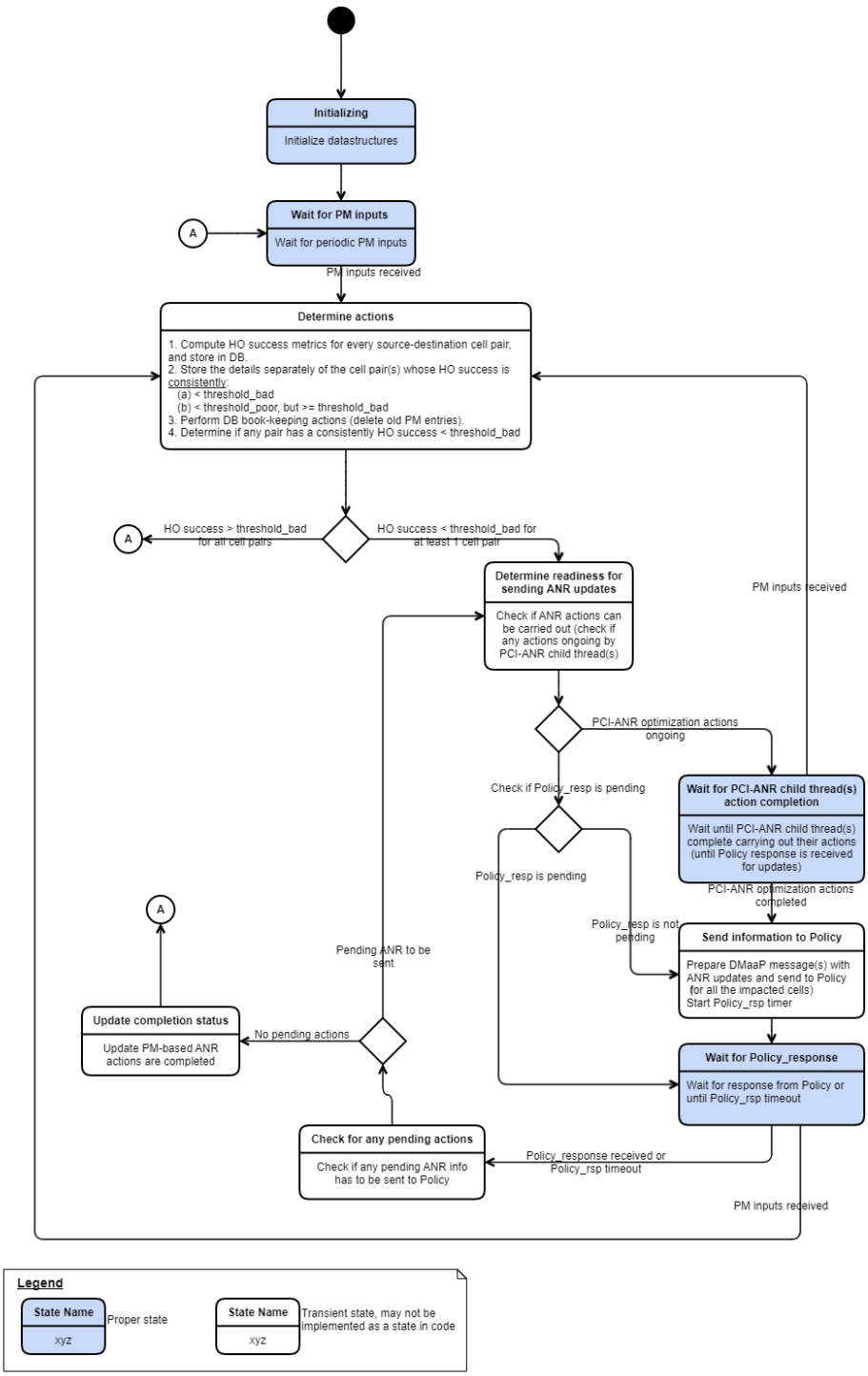
else

- Inform main thread of action completion

fi

2.3. PM-child thread

This child thread perform all PM-related computations, and initiate autonomous ANR updates based on HO metrics.



2.3.1. Initialization

In this state, all data structures are initialized. This includes any processing-pending PM data, processed PM data and populating details from DB.

2.3.2. Wait for PM inputs

In this state PM inputs are awaited. Upon reception of PM inputs, do the following:

2.3.2.1 Handle PM inputs

- Compute HO success metrics for every source-destination cell pair, and store in DB.
- Store the details separately of the cell pair(s) whose HO success is consistently:

a. < threshold_bad
b. < threshold_poor, but >= threshold_bad

- Perform DB book-keeping actions (delete old PM entries).
- if any pair has a consistently HO success < threshold_bad

then

2.3.2.2. Check readiness for ANR actions and carry out next steps

if ANR actions can be carried out (i.e., any PCI-ANR actions are ongoing by a PCI-ANR child thread)
then

if Policy_rsp is not pending

then

2.3.2.2.1. Prepare and send message(s) to Policy

Prepare and send DMaaP messages to Policy for all the cells whose neighbors have to be updated with "HO prohibited".

Start policy_rsp timer

Go to "Wait for Policy response" state (Section 2.3.4)

else

Go to "Wait for Policy response" state (Section 2.3.4)

fi

else

Go to "Wait for PCI-ANR child thread(s) action completion" state (Section 2.3.3)

fi

else

Remain in "Wait for PM inputs" state (Section 2.3.2).

fi

2.3.3. Wait for PCI-ANR child thread(s) action completion

When an indication that PCI-ANR child thread(s)' actions are completed is received, go to Step 2.3.2.2.1. If any PM inputs are received, go to Step 2.3.2.

2.3.4. Handling Policy response

- Response from policy is handled by a separate thread.
- When positive acknowledgement is received from the policy, the cells in the response are removed from the table and when negative acknowledgement is received the negative_ack count for the cell is increased and updated in the table. When no response is received no changes are made.
- When negative_ack count of a particular cell increases beyond a threshold, it is shifted to another table.