***Contents***

[***1 Chức năng kết nối đến MSC 3***](#_Toc95838537)

[***1.1 Thông tin chung chức năng 3***](#_Toc95838538)

[***1.2 Sự kiện kích hoạt 3***](#_Toc95838539)

[***1.3 Lưu đồ xử lí trong khối 4***](#_Toc95838540)

[***1.4 Thiết kế xử lý trong khối 4***](#_Toc95838541)

[***2 Chức năng quản lý trạng thái của MSC 8***](#_Toc95838542)

[***2.1 Thông tin chung chức năng 8***](#_Toc95838543)

[***2.2 Sự kiện kích hoạt 8***](#_Toc95838544)

[***2.3 Lưu đồ xử lý 8***](#_Toc95838545)

[***2.4 Thiết kế xử lý trong khối 8***](#_Toc95838546)

[***3 Chức năng thêm/xoá NRI mapping 11***](#_Toc95838547)

[***3.1 Thông tin chung chức năng 11***](#_Toc95838548)

[***3.2 Sự kiện kích hoạt 11***](#_Toc95838549)

[***3.3 Lưu đồ xử lý 11***](#_Toc95838550)

[***3.4 Thiết kế xử lý trong khối 12***](#_Toc95838551)

[***4 Chức năng nhận và xử lí bản tin từ node nghiệp vụ gửi đến MSC. 13***](#_Toc95838552)

[***4.1 Thông tin chung chức năng 13***](#_Toc95838553)

[***4.2 Sự kiện kích hoạt 13***](#_Toc95838554)

[***4.3 Lưu đồ xử lí 13***](#_Toc95838555)

[***4.4 Thiết kế xử lý trong khối 14***](#_Toc95838556)

[***5 Chức năng nhận và xử lí bản tin từ MSC gửi đến node nghiệp vụ. 16***](#_Toc95838557)

[***5.1 Thông tin chung chức năng 16***](#_Toc95838558)

[***5.2 Sự kiện kích hoạt 16***](#_Toc95838559)

[***5.3 Lưu đồ xử lí 16***](#_Toc95838560)

[***5.4 Thiết kế xử lí trong khối 17***](#_Toc95838561)

[***6 Chức năng lựa chọn MSC. 18***](#_Toc95838562)

[***6.1 Thông tin chung chức năng 18***](#_Toc95838563)

[***6.2 Sự kiện kích hoạt 18***](#_Toc95838564)

[***6.3 Lưu đồ xử lí 19***](#_Toc95838565)

[***6.4 Thiết kế xử lí trong khối 20***](#_Toc95838566)

[***7 Chức năng overload control 21***](#_Toc95838567)

[***7.1 Thông tin chung chức năng 21***](#_Toc95838568)

[***7.2 Sự kiện kích hoạt 21***](#_Toc95838569)

[***7.3 Lưu đồ xử lí trong khối 21***](#_Toc95838570)

[***7.4 Thiết kế xử lí trong khối 23***](#_Toc95838571)

# Chức năng kết nối đến MSC

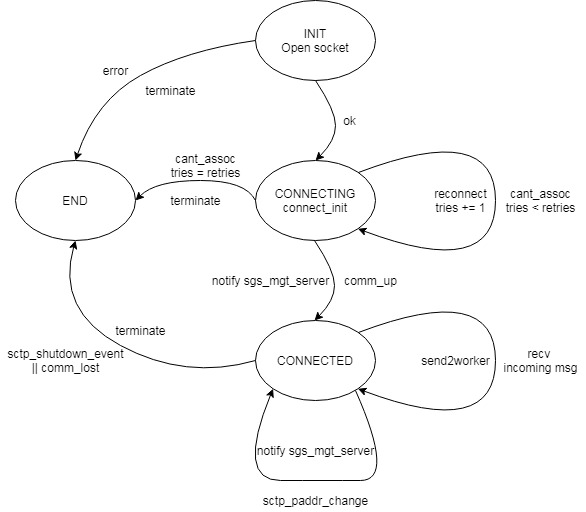
## Thông tin chung chức năng

* Đảm nhận việc khởi tạo, duy trì và đóng kết nối SCTP cho giao diện SGs đến các node MSC đầu xa.

## Sự kiện kích hoạt

* Khi node chuyển trạng thái từ backup sang active hoặc khi có thay đổi (thêm mới, sửa thông tin) cấu hình MSC trên NETCONF.

## Lưu đồ xử lí trong khối



Hình 1‑2 Sơ đồ chuyển trạng thái của kết nối đến MSC

## Thiết kế xử lý trong khối

Chức năng khởi tạo kết nối đến MSC được implement trên 2 module:

* ***msc\_connection\_sup***: là supervisor đảm nhận việc khởi tạo các tiến trình msc\_connector, khi node chuyển trạng thái từ backup -> master hoặc nhận được event thay đổi cấu hình MSC từ NETCONF. Ngoài ra khi một msc\_connector bất kỳ terminate, tiến trình msc\_connection\_sup sẽ thực hiện restart msc\_connector đó.
* ***msc\_connector***: đảm nhận việc khởi tạo kết nối SCTP đến các node MSC

***msc\_connection\_sup*** khởi tạo các ***msc\_connector*** với các tham số truyền vào tương ứng với các tham số cấu hình của MSC đầu xa mà tiến trình msc\_connector đó đảm nhận việc kết nối. Cụ thể bao gồm (*MSC name, MSC address, MSC port, MME port, MSC GId*).

1. Tiến trình ***msc\_connector*** được khởi tạo với trạng thái ban đầu là INIT với các tham số đầu vào được truyền từ ***msc\_connection\_sup*** (*MSC name, MSC address, MSC port, MME port, MSC GId*) và cấu hình SCTP nhận được từ NETCONF được lưu trong bảng ***mme\_conf***.

Trước hết thực hiện mở SCTP socket theo các tham số cấu hình SCTP:

* IP: là primary IP của giao diện SGs, trong trường hợp multi-homing thì bao gồm cả các secondary IP. Giá trị primary IP và secondary IP được lưu trong ***mme\_conf*** với key lần lượt là *mme\_sgs\_prim\_ip* và *mme\_sgs\_sec\_ips*.
* MME port: port number dùng cho việc mở socket, được truyền vào từ ***msc\_connection\_sup***
* Unordered: tham số quy định việc có sử dụng unordered delivery hay không, được lưu trong ***mme\_conf***.
* instream/ outstream: tương ứng với các giá trị max-instream và outstream của SCTP socket, được lưu trong ***mme\_conf***

Nếu socket được mở thành công thì lưu các thông tin trạng thái bao gồm:

* *msc\_ip*: lưu địa chỉ IP của MSC, có giá trị trùng với tham số đầu vào *MSC address*
* *ports*: lưu thông tin port kết nối của MME và MSC, là tuple có dạng {*MME port, MSC port*}.
* *connector*: lưu định danh socket vừa được mở thành công.
* peer\_name: lưu giá trị MSC name.
* *msc\_gid*: lưu giá trị *MSC GId*.
* *retry\_option*: lưu các cấu hình liên quan đến việc retry việc khởi tạo kết nối đến đầu xa, được cấu hình trên NETCONF, bao gồm các giá trị:
  + *delay*: thời gian delay việc retry ban đầu
  + *backoff* : hệ số mũ để tính toán khoảng thời gian delay sau những lần retry liên tục
  + *retry\_threshold*: số lần retry thất bại liên tục để bắn cảnh báo và terminate tiến trình.
  + *retry\_counter*: số lần retry thất bại liên tục. Giá trị ban đầu bằng 0.

rồi chuyển sang trạng thái CONNECTING; nếu không thì thực hiện terminate. Sau đó msc\_connection\_sup sẽ thực hiện bật tiến trình mới với các tham số truyền vào của tiến trình vừa bị terminate và tiến trình mới thực hiện INIT lại.

1. Khi chuyển trạng thái sang CONNECTING, tiến trình *msc\_connector* thực hiện thiết lập kết nối đến đầu xa bằng hàm ***gen\_sctp:connect\_init*** với các tham số được lưu trong trạng thái tiến trình khi INIT thành công:

* *socket*: là socket được mở khi INIT thành công.
* *MSC IP*: địa chỉ IP của MSC.
* *MSC port*: port của MSC

và đợi event trả về:

* Nếu ***gen\_sctp:connect\_init*** trả về *cant\_assoc* thì tính toán khoảng thời gian delay *T* dựa trên các tham số trong *retry\_option* được lưu trong state của tiến trình theo công thức:

*T­delay = InitialDelay \* round(retry\_counter backoff)*

và tăng giá trị retry\_counter lên 1.

Sau khoảng thời gian T thực hiện thiết lập lại kết nối bằng hàm ***gen\_sctp:connect\_init.***

* Nếu **gen\_sctp:connect\_init** trả về comm\_up:
  + Kiểm tra xem thông tin của MSC được lưu trong bảng *msc\_mgt\_table* tương ứng ( tên của bảng này chính là giá trị *MSC GId* được lưu trong trạng thái của tiến trình).
    - Nếu không tìm thấy bản ghi thì MSC được kết nối thành công lần đầu tiên => Insert các thông tin của SCTP association ứng với kết nối đến MSC đó vào bảng *msc\_mgt\_table* tương ứng. Các thông tin bao gồm:
      * *msc\_status*: nhận giá trị = 1 cho biết MSC đã ở trạng thái UP và sẵn sàng phục vụ.
      * *msc\_name*: định danh MSC
      * *assoc\_id*: định danh SCTP association.
      * *socket*: định danh socket kết nối.
      * *maxStreamId*: mặc định nhận giá trị -1 (???)
      * *msc\_addr*: địa chỉ IP của MSC
      * *msc\_port*: port number của MSC
      * *mme\_addr*: địa chỉ IP của MME
      * *mme\_port*: port number của MME
      * *instreams*: giá trị instream của SCTP association.
      * *outstreams*: giá trị outstream của SCTP association.
      * *is\_stable\_down*: cho biết MSC có ở trạng thái stable\_down hay không ( MSC ở trạng thái stable\_down khi mất kết nối SCTP trong khoảng thời gian vượt quá giá trị cấu hình ***stable\_down\_time*** ). Ở đây cờ này nhận giá trị false.

Gửi message thông báo kết nối thành công đến MSC (cho tiến trình *sgs\_mgt\_server*) với định dạng ***{msc\_up, msc\_name, msc\_gid}*** để chuyển trạng thái MSC sang STABLE\_UP ngay lập tức thay vì đợi khoảng thời gian *stable\_up\_time*. Việc cập nhật trạng thái MSC được xử lí bởi tiến trình *sgs\_mgt\_server. (Xem* [*phần 2*](#_2._Chức_năng)*)*

* + - Nếu tìm thấy bản ghi thì kiểm tra cờ *is\_stable\_down* trong bản ghi đó.
      * Nếu *is\_stable\_down* = true: MSC đang ở trạng thái STABLE\_DOWN.
        + cập nhật lại bản ghi với thông tin socket, assoc\_id, instreams, outstreams mới
        + msc\_status = 0 (MSC vẫn ở trạng thái DOWN và chỉ được chuyển sang trạng thái UP nếu kết nối được duy trì trong khoảng thời gian *stable\_up\_time*)
        + is\_stable\_down: nhận giá trị true

Gửi message thông báo cho tiến trình sgs\_mgt\_server với định dạng **{msc\_status\_notify, msc\_name, msc\_gid}** đến tiến trình sgs\_mgt\_server. (Xem [*phần 2*](#_Chức_năng_quản))

* + - * Nếu is\_stable\_down = false: Trường hợp này kết nối SCTP bị mất sau đó có lại sau khoảng thời gian nhỏ hơn tham số cấu hình ***stable\_down\_time.***
        + cập nhật lại bản ghi với thông tin socket, assoc\_id, instreams, outstreams mới
        + msc\_status = 1 (MSC vẫn ở trạng thái UP)
        + is\_stable\_down: nhận giá trị false
  + Gửi message resolve các alarm MSC\_CONNECT\_FAILURE và MSC\_SHUTDOWN trên EMS.
  + Chuyển sang trạng thái CONNECTED

1. Ở trạng thái CONNECTED, kết nối SCTP đến MSC đã được thiết lập thành công và sẵn sàng cho việc nhận và xử lí các bản tin SCTP gửi đến từ đầu xa.

Nếu nhận được event *sctp\_shutdown\_event* (khi nhận được SHUTDOWN trunk từ MSC) hoặc *sctp\_assoc\_change* với state = *comm\_lost* (khi quá nhiều bản tin bị timeout và retransmit):

* Đóng socket
* Raise alarm MSC\_SHUTDOWN (với trường hợp nhận được *sctp\_shutdown\_event* ) hoặc MSC\_CONNECT\_FAILURE( với event *sctp\_assoc\_change* với state = *comm\_lost* )
* Xoá mapping trong bảng *sgs\_sctp\_assoc\_table* có ứng với MSC để các bản tin nghiệp vụ không được gửi đến MSC đó nữa.

và terminate tiến trình. Sau đó *msc\_connection\_sup* sẽ thực hiện bật tiến trình mới với các tham số truyền vào của tiến trình vừa bị terminate và tiến trình mới thực hiện INIT lại.

# Chức năng quản lý trạng thái của MSC

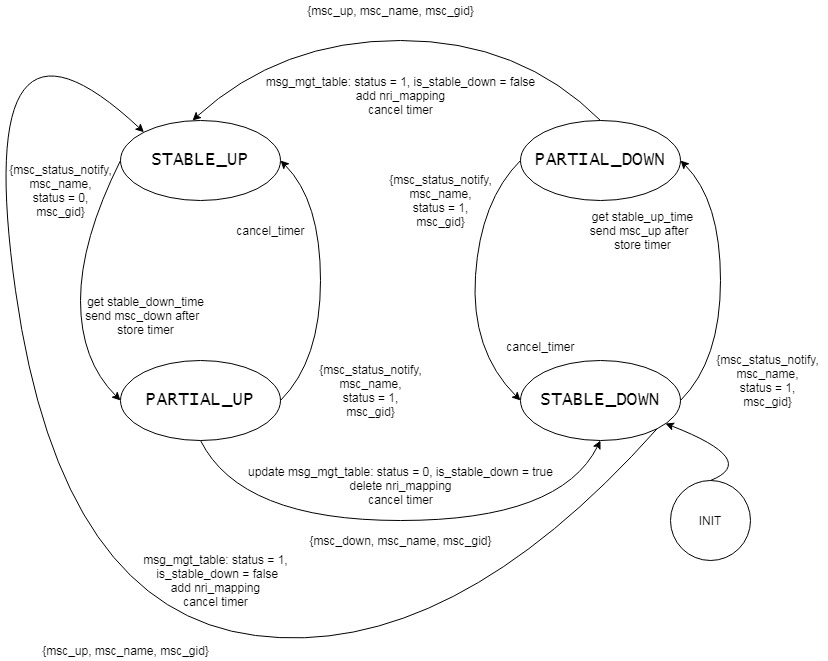
## Thông tin chung chức năng

* Đảm nhận việc quản lý và thay đổi trạng thái của MSC, từ đó quyết định việc có cho phép gửi nhận bản tin nghiệp vụ đến MSC đó hay không.

## Sự kiện kích hoạt

* Khi nhận được bản tin thông báo việc thay đổi trạng thái của kết nối SCTP ứng với MSC đó.

## Lưu đồ xử lý



Hình 2‑1 Sơ đồ chuyển trạng thái MSC

## Thiết kế xử lý trong khối

* Tính năng quản lý trạng thái MSC được implement trên module ***sgs\_mgt\_server***.
* Các trạng thái của MSC trên sgs\_gw bao gồm:
* STABLE\_UP: Kết nối SCTP đến MSC được thiết lập thành công và đã tồn tại trong khoảng thời gian nhiều hơn giá trị ***stable\_up\_time***. Các bản tin nghiệp vụ được gửi nhận bình thường giữa MME và MSC và giá trị *msc\_status* trong bản ghi ứng với MSC của bảng msc\_mgt\_table = 1.
* PARTIAL\_UP: Kết nối SCTP đến MSC đã bị mất nhưng khoảng thời gian mất kết nối chưa vượt quá giá trị ***stable\_down\_time*.** Giá trị *msc\_status* trong bản ghi ứng với MSC của bảng msc\_mgt\_table vẫn = 1.
* STABLE\_DOWN: Kết nối SCTP đến MSC đã mất trong khoảng thời gian dài hơn giá trị ***stable\_down\_time***. Giá trị *msc\_status* trong bản ghi ứng với MSC của bảng msc\_mgt\_table = 0.
* PARTIAL\_DOWN: Kết nối SCTP đến MSC đã được thiết lập lại sau khi bị mất trước đó nhưng khoảng thời gian kể từ lúc kết nối thành công chưa vượt quá giá trị ***stable\_up\_time*.** Giá trị *msc\_status* trong bản ghi ứng với MSC của bảng msc\_mgt\_table vẫn = 0.
* Luồng xử lí các event nhận được từ các tiến trình ***msc\_connector*** trên ***sgs\_mgt\_server***:
* Ở trạng thái STABLE\_UP:
  + Nhận được bản tin ***{msc\_status\_notify, msc\_name, status, msc\_gid}*** với status = 0: Kết nối SCTP đến MSC đã bị mất, tiến trình thực hiện get tham số cấu hình *stable\_down­\_time* từ bảng *mme\_conf,* sau đó dùng hàm ***erlang:send\_after*** để gửi bản tin có định dạng {msc\_down, msc\_name, msc\_gid} đến chính nó sau khoảng thời gian *stable\_down\_time* . Cuối cùng lưu timer được trả về từ hàm ***erlang:send\_after*** và chuyển sang trạng thái PARTIAL\_UP.
* Ở trạng thái PARTIAL\_UP:
  + Nếu nhận được bản tin ***{msc\_status\_notify, msc\_name, status = 1, msc\_gid}*** chứng tỏ kết nối SCTP đã được khôi phục, thực hiện huỷ timer (để huỷ việc gửi bản tin msc\_down ứng với timer đó) và chuyển về trạng thái STABLE\_UP.
  + Nếu nhận được bản tin {msc\_down, msc\_name, msc\_gid}, nghĩa là sau khoảng thời gian stable\_down\_time kết nối SCTP không được khôi phục lại: Thực hiện cập nhật trạng thái của MSC trong bảng msc\_mgt\_table tương ứng với status = 0, is\_stable\_down = true; sau đó xoá các NRI mapping của MSC đó trong bảng NRI mapping, huỷ timer trước đó (nếu có) và chuyển sang trạng thái STABLE\_DOWN.
* Ở trạng thái STABLE\_DOWN:
  + Nhận được bản tin ***{msc\_up, msc\_name, msc\_gid}*** : kết nối thành công đến MSC lần đầu tiên => Thực hiện cập nhật trạng thái của MSC trong bảng *msc\_mgt\_table* tương ứng với *status = 1*, *is\_stable\_down = false*; sau đó thêm NRI mapping của MSC đó trong bảng NRI mapping, huỷ timer trước đó (nếu có) và chuyển sang trạng thái STABLE\_UP.
  + Nhận được bản tin ***{msc\_status\_notify, msc\_name, status = 1, msc\_gid}*** với cho biết kết nối SCTP đến MSC bị mất trước đó đã được khôi phục, tiến trình thực hiện get tham số cấu hình *stable\_up­\_time* từ bảng *mme\_conf,* sau đó dùng hàm ***erlang:send\_after*** để gửi bản tin có định dạng ***{msc\_up, msc\_name, msc\_gid}*** đến chính nó sau khoảng thời gian *stable\_up\_time* . Cuối cùng lưu timer được trả về từ hàm ***erlang:send\_after*** và chuyển sang trạng thái PARTIAL\_DOWN.
* Ở trạng thái PARTIAL\_DOWN:
  + Nếu nhận được bản tin ***{msc\_status\_notify, msc\_name, status = 0, msc\_gid}*** chứng tỏ kết nối SCTP vừa được khôi phục đã bị mất, thực hiện huỷ timer (để huỷ việc gửi bản tin msc\_up ứng với timer đó) và chuyển về trạng thái STABLE\_DOWN.
  + Nếu nhận được bản tin ***{msc\_up, msc\_name, msc\_gid}***, nghĩa là sau khoảng thời gian *stable\_down\_time* kết nối SCTP không bị mất => thực hiện cập nhật trạng thái và thêm NRI mapping như đã trình bày ở trạng thái STABLE\_DOWN.

# Chức năng thêm/xoá NRI mapping

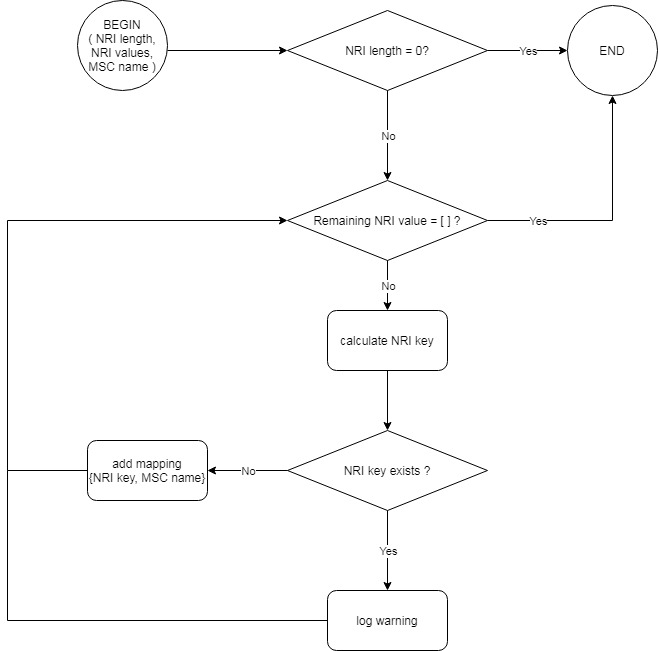
## Thông tin chung chức năng

* Thực hiện việc thêm/xoá mapping giữa NRI key với định danh MSC dùng cho việc lựa chọn MSC.

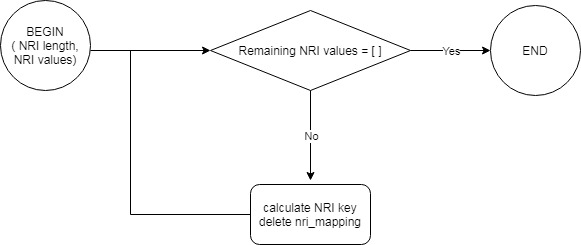
## Sự kiện kích hoạt

* Khi kết nối đến MSC chuyển sang trạng thái STABLE\_UP hoặc STABLE\_DOWN

## Lưu đồ xử lý



Hình 3‑1 Lưu đồ xử lí luồng add\_nri\_mapping



Hình 3‑2 Lưu đồ xử lí luồng delete\_nri\_mapping.

## Thiết kế xử lý trong khối

NRI mapping là mapping giữa *{NRI key, MSC name}* được sử dụng trong luồng lựa chọn MSC dựa theo giá trị TMSI. Giá trị NRI key dựa trên NRI length và NRI value theo công thức:

NRI key = NRI length \* 1000 + NRI value

* Việc thêm NRI mapping được thực hiện khi MSC chuyển trạng thái sang STABLE\_UP, và được implement trong hàm *add\_nri\_mapping* với 3 tham số:
* NRI length: giá trị cấu hình NRI length của MSC được lưu trong mme\_conf
* List NRI value: danh sách các NRI value của MSC được lưu trong mme\_conf
* MSC name: tên MSC.

Luồng xử lí được thực hiện qua các bước:

* Bước 1: Kiểm tra giá trị NRI length được truyền vào; nếu NRI length = 0 thì kết thúc luồng, nếu NRI length > 0 thì chuyển đến bước 2.
* Bước 2: Chạy vòng lặp qua List NRI value:
* Tính giá trị NRI key dựa trên NRI length và NRI value theo công thức:

NRI key = NRI length \* 1000 + NRI value

* Lookup giá trị NRI key vừa tính được trong bảng nri\_mapping:
* Nếu tồn tại bản ghi: log cảnh báo
* Nếu không tồn tại bản ghi: Insert mapping *{NRI key, MSC name}* vào bảng nri\_mapping
* Tiếp tục đến khi duyệt hết các giá trị NRI value thì kết thúc luồng.
* Việc xoá NRI mapping được thực hiện khi MSC chuyển trạng thái sang STABLE\_DOWN, và được implement trong hàm *delete\_nri\_mapping* với 2 tham số:
* NRI length: giá trị cấu hình NRI length của MSC được lưu trong mme\_conf
* List NRI value: danh sách các NRI value của MSC được lưu trong mme\_conf.

Luồng xử lí được thực hiện qua các bước:

* Chạy vòng lặp qua List NRI value:
* Tính toán giá trị NRI key
* Xoá mapping trong bảng nri\_mapping dựa theo giá trị NRI key tìm được.
* Sau khi duyệt qua tất cả các phần tử trong List NRI value, kết thúc luồng.

# Chức năng nhận và xử lí bản tin từ node nghiệp vụ gửi đến MSC.

## Thông tin chung chức năng

* Đảm nhận việc gửi bản tin giữa các node nghiệp vụ đến các node MSC đầu xa.

## Sự kiện kích hoạt

* Khi nhận được bản tin từ node nghiệp vụ hoặc MSC đầu xa.

## Lưu đồ xử lí

## 

Hình 4‑1 Lưu đồ xử lí bản tin từ nghiệp vụ gửi đến MSC

## Thiết kế xử lý trong khối

* Các bản tin nghiệp vụ của giao diện SGs được gửi/nhận giữa các node nghiệp vụ và sgs\_gw dưới dạng record như sau:

#sgsap\_relay\_msg {

*socket*, // socket nhận trong trường hợp bản tin là phản hồi cho request từ MSC.

*assoc\_id*, // assoc\_id định danh association ứng với kết nối đến MSC.

*stream*, // stream ID định danh association ứng với kết nối đến MSC.

*msc\_gid*, // MSC group ID

*msc\_name*, // Tên MSC

imsi,

*msg\_type*, // message type theo chuẩn 3GPP 29.118

*payload*, // chuỗi binary chứa bản tin đã được encode

*tmsi\_nri*, // giá trị TMSI dùng cho việc lựa chọn MSC

*partition\_id* // partion ID ứng với giá trị IMSI

}

* Ngoài các bản tin nghiệp vụ của giao diện SGs còn có các bản tin với định dạng *{free\_msc\_assoc, IMSI}* nhằm mục đích xoá mapping giữa IMSI với thông tin kết nối SCTP của MSC.
* Thông tin mapping giữa IMSI với kết nối SCTP ứng với MSC đầu xa được lưu trong bảng sgs\_sctp\_assoc\_table có định dạng như sau:

#sgs\_sctp\_assoc\_table {

imsi,

assoc\_id,

socket,

streamID,

msc\_name,

msc\_addr

}

* Thông tin mapping giữa IMSI với partition ID của partition chứa các node phục vụ IMSI được lưu trong bảng imsi\_partition\_id có định dạng *{IMSI, partition ID}.*
* Các bước xử lí:
* Bước 1: Kiểm tra bản tin đến có phải là bản tin *{free\_msc\_assoc, IMSI}* không; nếu có thì thực hiện xoá mapping IMSI-partition ID ứng với IMSI trong bản tin, nếu không thì chuyển sang bước 2.
* Bước 2: Kiểm tra xem node có đang ở trạng thái overload chiều outbound (gửi bản tin từ nghiệp vụ đến đầu xa) hay không bằng cách kiểm tra giá trị của cờ OUTBOUND\_OVERLOAD\_FLAG; nếu nhận giá trị **true** thì discard bản tin và kết thúc luồng, nếu không thì chuyển sang bước 3.
* Bước 3: Kiểm tra trường IMSI trong bản tin nhận được; nếu IMSI = NULL thì log cảnh báo và thực hiện gửi bản tin dựa theo các trường {socket, assoc\_id, stream, payload} trong bản tin nếu có thể, nếu IMSI ≠ NULL thì chuyển sang bước 4.
* Bước 4: Update mapping giữa IMSI với partition ID dựa vào các trường tương ứng trong bản tin nhận được và chuyển sang bước 5.
* Bước 5: Kiểm tra xem IMSI trong bản tin đã tồn tại trong bảng *sgs\_sctp\_assoc\_table* hay chưa; nếu đã tồn tại thì dùng các thông tin {socket, assoc\_id, streamID} trong mapping để gửi payload trong bản tin đếu đầu xa tương ứng và kết thúc luồng, nếu không thì chuyển sang bước 6.
* Bước 6: Kiểm tra xem có MSC nào đang ở trạng thái stable và sẵn sàng gửi nhận bản tin hay không; nếu không thì discard bản tin và kết thúc luồng, nếu có thì chuyển sang bước 7.
* Bước 7: Lựa chọn một trong số các MSC đang ở trạng thái stable để gửi bản tin đến dựa theo giá trị TMSI\_NRI trong bản tin và nri\_mapping ứng với các MSC.

# Chức năng nhận và xử lí bản tin từ MSC gửi đến node nghiệp vụ.

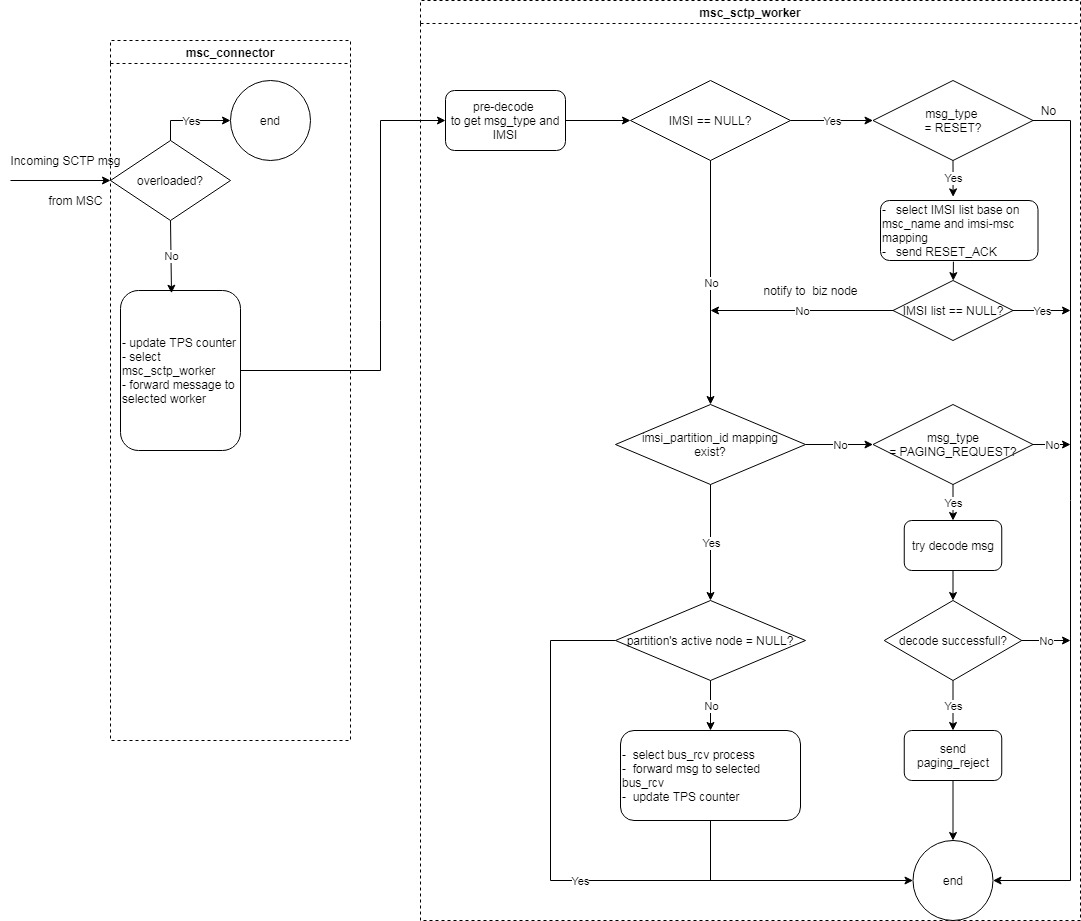
## Thông tin chung chức năng

* Đảm nhận việc nhận bản tin từ MSC, decode và gửi đến node nghiệp vụ.

## Sự kiện kích hoạt

* Khi nhận được bản tin SCTP từ MSC gửi sang

## Lưu đồ xử lí



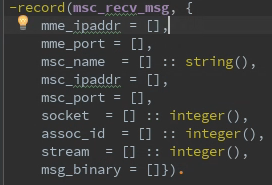
Hình 5‑1 Lưu đồ xử lí bản tin gửi từ MSC đến nghiệp vụ

## Thiết kế xử lí trong khối

Việc xử lí bản tin nhận được từ MSC được implement trên 2 khối:

* ***msc\_connector***: đảm nhận việc nhận bản tin từ MSC và forward đến các tiến trình msc\_sctp\_worker.
* ***msc\_sctp\_worker***: đảm nhận việc tiền xử lí bản tin để lựa chọn và forward bản tin đến node nghiệp vụ.

Định dạng bản tin giao tiếp giữa msc\_connector và msc\_sctp\_worker:



Định dạng bản tin giao tiếp giữa ***msc\_sctp\_worker*** với node nghiệp vụ được định nghĩa ở mục 3.

Khi nhận được bản tin từ MSC gửi tới, ***msc\_connector*** thực hiện:

* Check trạng thái inbound overload: Nếu đang ở trạng thái overload -> thực hiện discard bản tin. Nếu không thì chuyển sang bước tiếp theo.
* Lựa chọn worker để forward bản tin theo cơ chế round-robin như sau:
* Lấy thông tin worker\_id từ trong state (giá trị khởi tạo của worker\_id = 1) và convert thành worker\_name, vi id = 1 -> worker\_name = msc\_sctp\_worker\_1
* Gửi bản tin đến tiến trình msc\_sctp\_worker được chọn theo định dạng record msc\_recv\_msg như trên.
* Update lại giá trị worker\_id theo công thức sau để đảm bảo round đều giữa các worker:

*NewWorkerId = WorkerId rem NumberOfWorker + 1*

Khi nhận được bản tin từ msc\_connector gửi đến, msc\_sctp\_worker thực hiện:

* Decode sơ bản tin để lấy giá trị msg\_type và IMSI.
* Nếu giá trị IMSI nhận được = [ ], thực hiện check msg\_type.
* Nếu msg\_type cho biết bản tin nhận được là bản tin SGsAP\_RESET\_INDICATION thì thực hiện gửi bản tin RESET\_ACK ngay lập tức, sau đó:
* Select danh sách các IMSI đang được phục vụ bởi MSC dựa trên bảng mapping sgs\_sctp\_assoc\_table và trường msc\_name trong bản tin
* Nếu danh sách IMSI = NULL: kết thúc luồng
* Nếu danh sách IMSI != NULL: với mỗi IMSI gửi binary payload trong trường msg\_binary đến các node nghiệp vụ đang phục vụ IMSI đó
* Nếu không thì drop bản tin và kết thúc luồng.
* Nếu IMSI != NULL: thực hiện lựa chọn node nghiệp vụ và forward bản tin theo các bước:
* Lookup mapping giữa IMSI với partition ID định danh cho partition chứa các node phục vụ IMSI đó (mapping này được lưu khi xử lí bản tin từ nghiệp vụ gửi đến đầu xa trong [mục 3](#_Chức_năng_nhận) ).
* Nếu không tìm thấy bản tin thì thực hiện check msg\_type có phải PAGING\_REQUEST hay không:
* Nếu không thì kết thúc luồng
* Nếu có thì thực hiện decode bản tin; nếu decode thành công thì tạo và gửi bản tin PAGING\_REJECT đến MSC, nếu decode không thành công thì kết thúc luồng.
* Nếu tìm thấy bản ghi thì thực hiện kiểm tra xem có node nào thuộc partition tìm được đang UP hay không
* Nếu không tìm được thì log cảnh bảo và kết thúc luồng
* Nếu tìm được thì lựa chọn tiến trình sgs\_handler trên node đó theo cơ chế round\_robin, sau đó gửi bản tin đến cho tiến trình tìm được (theo định dạng record *sgsap\_relay\_message* ) rồi update giá trị counter trước khi kết thúc luồng.

# Chức năng lựa chọn MSC.

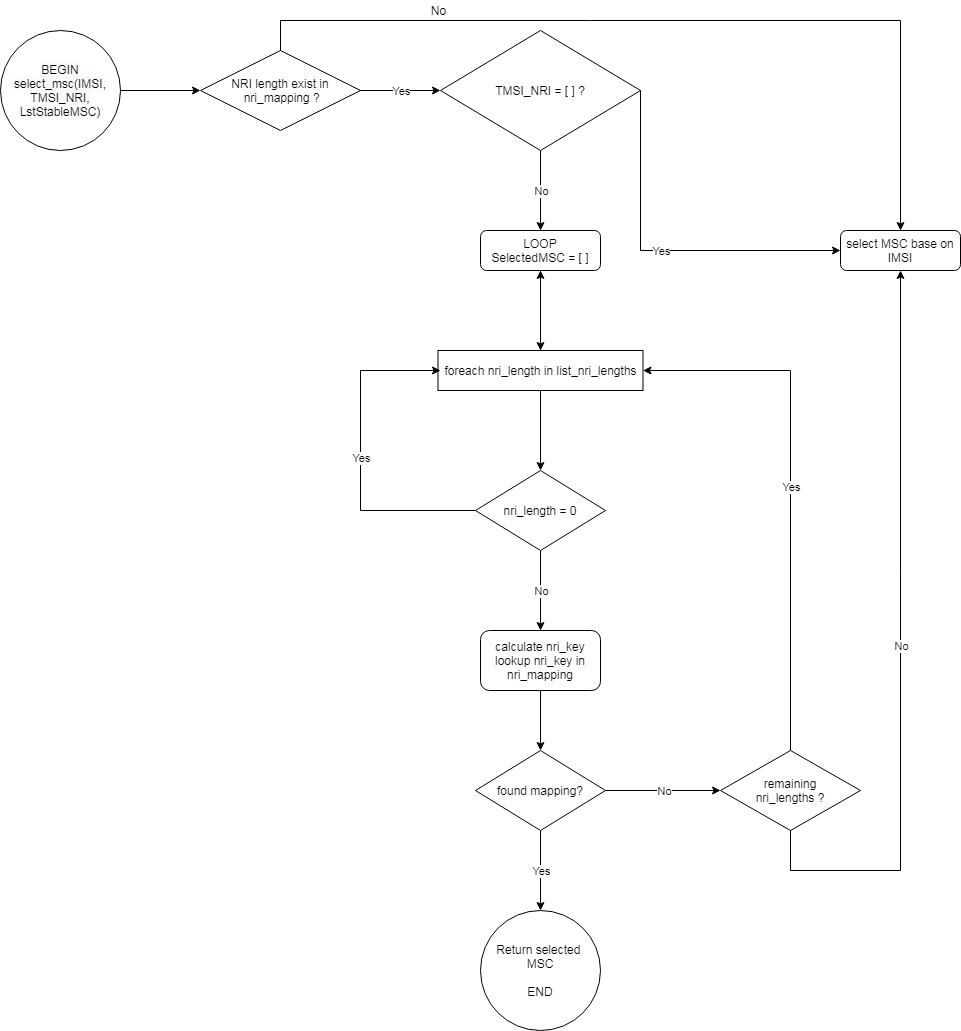
## Thông tin chung chức năng

* Đảm nhận việc lựa chọn MSC để forward bản tin trong trường hợp không có mapping *sgs\_sctp\_assoc\_table*.

## Sự kiện kích hoạt

* Khi nhận được bản tin từ node nghiệp vụ và không tìm thấy mapping giữa IMSI trong bản tin và SCTP association trong bảng *sgs\_sctp\_assoc\_table*.

## Lưu đồ xử lí



Hình 6‑1 Lưu đồ xử lí nghiệp vụ lựa chọn MSC

## Thiết kế xử lí trong khối

Các tham số đầu vào bao gồm:

* IMSI
* *TMSI\_NRI*: Giá trị TMSI\_NRI
* *LstStableMsc*: Danh sách các MSC đang ở trạng thái STABLE\_UP

Các bước xử lí:

1. Lookup giá trị *nri\_length* trong bảng *nri\_mapping* và kiểm tra giá trị TMSI\_NRI được truyển vào.

* Nếu kết quả lookup *nri\_length* = [] (trong trường hợp bất thường) hoặc TMSI\_NRI = [] => lựa chọn MSC bằng IMSI theo các bước:

+ Tính giá trị *I =*  số dư của phép chia *IMSI* chia cho độ dài của *LstStableMsc*

+ Trả về phần tử thứ *I* trong *LstStableMsc* và kết thúc luồng

* nếu không thì chuyển sang bước 2

1. Lấy giá trị *ListNriLength* chứa danh sách các giá trị *nri\_length* của các MSC từ kết quả lookup bên trên, sau thực hiện vòng lặp trên các phần tử trong list đó như sau:

* *nri\_length = 0* : do không có giá trị *nri\_value* nào ứng với giá trị *nri\_length = 0*, tiếp tục vòng lặp với nri\_length tiếp theo.
* *nri\_length != 0:* Tính toán giá trị *NriKey* theo công thức:



Lookup giá trị *NriKey* trong bảng *nri\_mapping:*

* + Nếu không tìm thấy bản ghi: bỏ qua và tiếp tục vòng lặp với nri\_length tiếp theo.
  + Nếu tìm thấy bản ghi chứa mapping giữa *NriKey* và MSC: trả về tên MSC và kết thúc luồng.

1. Nếu sau vòng lặp không tìm thấy MSC thoả mãn: thực hiện lựa chọn MSC bằng IMSI (như trong bước 1) và kết thúc luồng.

# Chức năng overload control

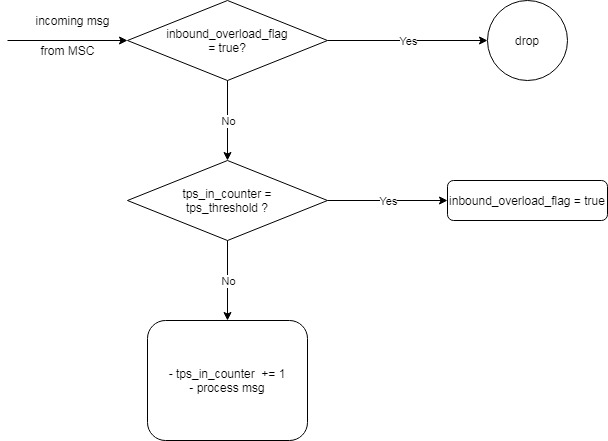
## Thông tin chung chức năng

Thực hiện drop các bản tin vượt ngưỡng TPS được cấu hình nhằm tránh gây hiện tượng quá tải cho node mạng đầu xa và node nghiệp vụ.

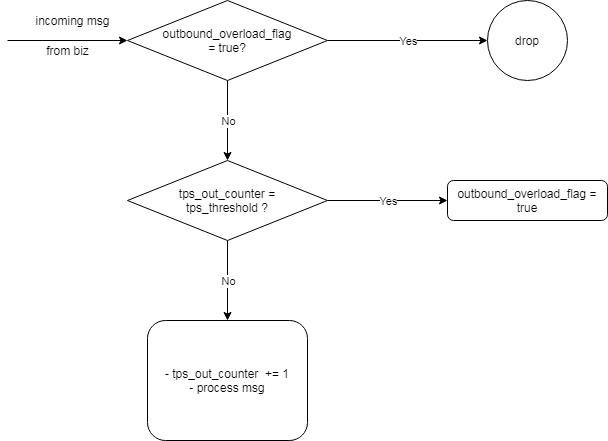
## Sự kiện kích hoạt

Nhận được bản tin SCTP từ MSC gửi đến hoặc bản tin *sgsap\_relay\_message* từ nghiệp vụ gửi sang.

## Lưu đồ xử lí trong khối



Hình 7‑1 Lưu đồ xử lí overload cho bản tin từ MSC gửi đến nghiệp vụ.



Hình 7‑2 Lưu đồ xử lí overload cho bản tin từ nghiệp vụ gửi đến MSC.

## Thiết kế xử lí trong khối

Việc xử lí overload cho bản tin gửi từ MSC đến node nghiệp vụ được thực hiện theo các bước:

* Bước 1: Kiểm tra giá trị cờ INBOUND\_OVERLOAD\_FLAG (giá trị khởi tạo của cờ này là false).
* Nếu INBOUND\_OVERLOAD\_FLAG = true -> node đang ở trạng thái overload, thực hiện drop bản tin
* Nếu INBOUND\_OVERLOAD\_FLAG = false -> node đang ở trạng thái bình thường, chuyển sang bước 2.
* Bước 2: So sánh giá trị *tps\_in\_counter* với giá trị cấu hình *tps\_in\_threshold*:
* Nếu *tps\_in\_counter* < *tps\_in\_threshold*: tăng giá trị tps\_in\_counter lên 1 và thực hiện xử lí bản tin.
* *Nếu tps\_in\_counter* = *tps\_in\_threshold*: giá trị TPS đã đạt ngưỡng, thực hiện set giá trị cờ INBOUND\_OVERLOAD\_FLAG = true. Khi này các bản tin vượt ngưỡng tiếp theo sẽ bị drop

Việc xử lí overload cho bản tin gửi từ nghiệp vụ đến MSC được thực hiện theo các bước:

* Bước 1: Kiểm tra giá trị cờ OUTBOUND\_OVERLOAD\_FLAG (giá trị khởi tạo của cờ này là false).
* Nếu OUTBOUND \_OVERLOAD\_FLAG = true -> node đang ở trạng thái overload, thực hiện drop bản tin
* Nếu OUTBOUND \_OVERLOAD\_FLAG = false -> node đang ở trạng thái bình thường, chuyển sang bước 2.
* Bước 2: So sánh giá trị *tps\_out\_counter* với giá trị cấu hình *tps\_out\_threshold*:
* Nếu *tps\_out\_counter* < *tps\_out\_threshold*: tăng giá trị *tps\_out\_counter* lên 1 và thực hiện xử lí bản tin.
* Nếu *tps\_out\_counter* = *tps\_out\_threshold*: giá trị TPS đã đạt ngưỡng, thực hiện set giá trị cờ OUTBOUND\_OVERLOAD\_FLAG = true. Khi này các bản tin vượt ngưỡng tiếp theo sẽ bị drop.