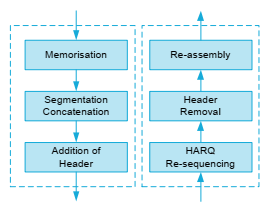
1. Các chức năng chính của RLC

* Vi lớp MAC không quan tâm tới phân mảnh bản tin và thứ tự bản tin, RLC sẽ phụ trách các nhiệm vụ này.
* Do đó, nhiệm vụ chính của RLC như sau:
  + Tăng độ tin cậy của việc trao đổi bản tin
  + Sắp xếp lại thứ tự bản tin
  + Quản lý việc phân mảnh dữ liệu
* Tuy nhiên, do có độ trễ nhất định nên tùy vào từng kiểu dữ liệu, các tác vụ này có thể tùy chọn để áp dụng.

1. Các mode trong RLC

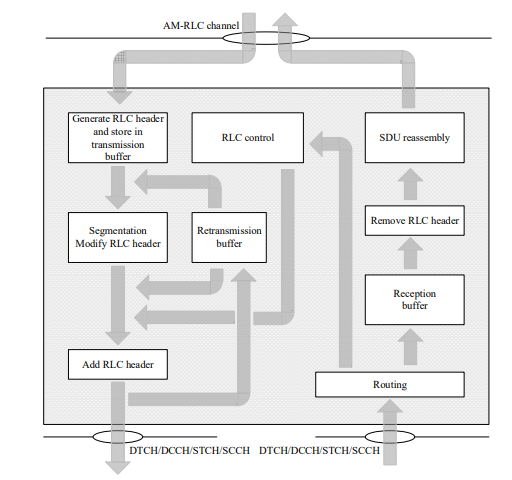
* Transpatent Mode (TM)
  + Là mode đơn giản nhất khi RLC không làm gì cả
  + Vì không có phân mảnh dữ liệu nên mode này phù hợp với các bản tin ngắn, chủ yếu là các bản tin signaling.
* Unacknowledgement Mode (UM)
  + Có khả năng sắp xếp lại thứ tự gói tin
  + Có khả năng phân mảnh và nối các gói dữ liệu dựa trên yêu cầu của lớp MAC.
  + Vì không sử dụng phản hồi nên không ảnh hưởng nhiều tới độ trễ, nhưng đồng thời cũng không tăng cường độ tin cậy. Vì thế mode này phù hợp với những ứng dụng thời gian thực như video call, streaming.
* Acknowledgement Mode (AM)
  + Tương tự như UM, nhưng có thêm khả năng truyền lại các gói bị mất.
  + Do sử dụng phản hồi, mode này tăng độ tin cậy nhưng cũng làm tăng độ trễ.
  + Do đó, mode này phù hợp với các dịch vụ như truyền file, duyệt web…

1. Cách thức hoạt động của UM và AM
   1. Cách thức hoạt động của UM



* Khi nhận được một gói từ lớp trên (RRC, PDCP,… ), RLC sẽ lưu trữ nó ở trong buffer trong khi đợi lớp MAC yêu cầu MAC-SDU 🡪 có buffer ở bên truyền.
* RLC sẽ xác định kích thước MAC-SDU mong muốn để thức hiện phân mảnh hoặc nối cho phù hợp
* Dể

RLC AM



**Ghi chú:**

* Khác với 4G, phần tử RLC phát không thực hiện việc ghép nối (concatenate)

**Giới thiệu**

* Trong 5G NR, một AM Data (AMD) PDU luôn luôn bao gồm một RLC header, lần lượt mang Sequence Number (SN) của RLC SDU.
* SDUs có thể bị phân mảnh (PDU thì không).
* SN và Segment Offset (SO) là các trường quan trọng trong header có thể giúp cho việc phản hồi và truyền lại.
* STATUS PDU là một loại control PDU và nó cũng có header.
* Các cách phản ứng được điều khiển bởi các biến trạng thái, hằng số và các bộ đếm.

**Kích thước cửa sổ** ở cả phía phát và phía thu (khoảng để kiểm tra SN có nằm trong):

* 0 – 2048: tương ứng với 12 bit SN
* 0 – 131072: tương ứng với 18 bit SN

**Các biến trạng thái** trong AM bao gồm: Tx\_Next\_Ack, Tx\_Next, Rx\_Next, Rx\_Next\_status\_trigger, Rx\_Next\_highest

Biến trạng thái ở phía PHÁT:

**Tx\_next**: Biến trạng thái Tx

* Biến trạng thái này thể hiện giá trị SN sẽ được gán với **AMD PDU** được tạo mới tiếp theo.
* Giá trị Tx\_next được khởi tạo bằng 0, sau đó sẽ được cập nhật mỗi khi RLC AM tạo ra một PDU AMD mới với SN = Tx\_next và chứa cả một RLC SDU

**Tx\_Next\_ACK**: biến trạng thái Tx ACK

* Thể hiện giá trị của SN của **RLC SDU** tiếp theo, theo đó phản hồi ACK cho các RLC SDUs cũng sẽ theo thứ tự lần lượt. (hay biến này thể hiện SN của SDU tiếp theo theo thứ tự mà cần một phản hồi tích cực – positive acknowledge).
* Giá trị SN của Tx\_next\_ack là giá trị biên thấp nhất của “transmitting window”.
* Ban đầu, biến trạng thái Tx\_next\_ack được đặt bằng 0, sau đó được cập nhật mỗi khi RLC AM nhận được một positive ACK cho RLC SDU với SN = Tx\_next\_ack
* Có thể nhận thấy TX\_Next\_Ack <= TX\_Next

**POLL\_SN:**

* Là giá trị SN cao nhất trong số các PDUs được chuyển tới lớp dưới ở thời điểm polling.
* Nó khởi tạo với giá trị ban đù là 0.

CÁC COUNTER Ở PHÍA PHÁT:

**PDU\_WITHOUT\_POLL:**

* Số PDU đựơc gửi kể từ lần poll gần nhất.
* Giá trị khởi tạo là 0.

**BYTE\_WITHOUT\_POLL:**

* Số bytes được gửi kể từ lần poll gần nhất.
* Giá trị khởi tạo là 0.

**RETX\_COUNT:**

* Số lần truyền lại của 1 SDU hay mảnh của nó.
* Mỗi RLC SDU có 1 RETX\_CONT riêng.

Liên quan đến các counters là **3 thông số** mà RRC cấu hình:

**MaxRetxThreshold**:

* Là giới hạn số lần truyền lại.
* Khi RETX\_COUNT chạm đến ngưỡng này, lớp trên sẽ báo truyền SDU bị lỗi. Nó có thể dẫn đến thủ tục tái lập RRC.

**PollPDU**:

* Khi PDU\_WITHOUT\_POLL chạm đến ngưỡng này, polling sẽ được bắt đầu.

**PollByte**:

* Khi BYTE\_WITHOUT\_POLL chạm đến ngưỡng này, polling sẽ được bắt đầu.

Biến trạng thái ở phía thu:

**RX\_Next:**

* Thể hiển biên dưới của cửa số nhận.
* Giá trị khởi tạ là 0.
* Được cập nhật khi một SDU với SN = RX\_Next được nhận hoàn toàn.

**RX\_Next\_Highest:**

* Chỉ SN tiếp theo sau SN lớn nhất trong số các SDUs nhận được, cho dù SDU với SN lớn nhất vẫn chưa được nhận hoàn toàn.
* Do đó, RX\_Next <= RX\_Next\_Highest.
* Gá trị khởi tạo là 0.

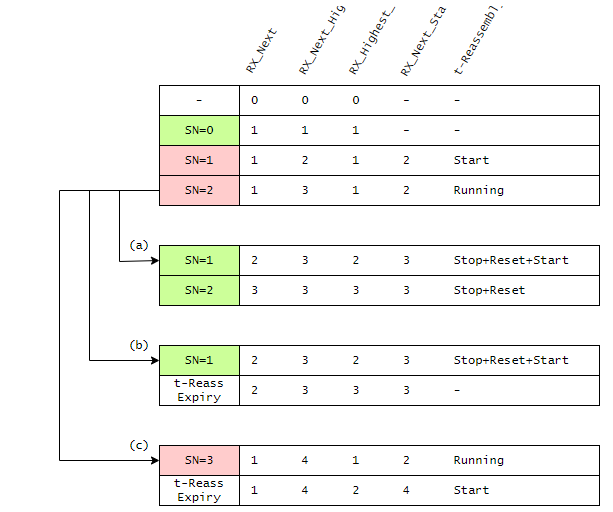
**RX\_Highest\_Status:**

* Là giá trị cao nhất có thể được thể hiện trong trường ACK\_SN của STATUS PDU.
* SDUs với SN thấp hơn mà chưa được nhận hoàn toàn sẽ được thể hiện với NACK\_SN trong STATUS PDU.
* Giá trị khởi tạo bằng 0.

**RX\_Next\_Status\_Trigger:**

* SN theo sau SN của RLC SDU kích hoạt t-Reassembly.
* Do đó, nếu SN = 2 được nhận một phần và t-Reassembly được bắt đầu cho nó, biến trạng thái này sẽ được đặt là 3.

MÔ TẢ CẬP NHẬT Biến trạng thái ở phía thu:



Hình 1. Thay đổi các biến trạng thái trong 3 trường hợp a, b, và c

Hình trên mô tả sự thay đổi của biến trạng thái trong 3 trường hợp. Trong cả 3 trường hợp, bắt đầu với SN = 0 được nhận hoàn toàn, sau đó SN = 1 và SN = 2 được nhận một phần.

Khi SN = 1 được nhận một phần, t-Reassambly được bắt đầu. Khi SN = 2 được nhận một phần, không có hành động nào với timer do t-Reassembly đã chạy cho SN = 1.

Trong trường hợp a:

* SN = 1 được nhận hoàn toàn. Do đó, t-Reassembly dừng lại và reset.
* Nhưng vì SN = 2đã được nhận 1 phần trước đó, t-Reassembly được bắt đầu cho SN = 2.
* Khi SN = 2được nhận hoàn toàn, reassembly hoàn thành cho SDU này và timer được dừng lại rồi reset.

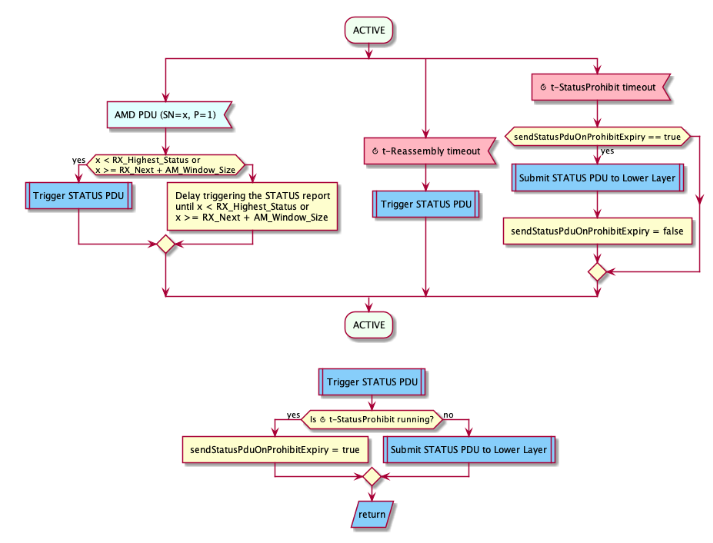
Trong trường hợp b:

* SN = 1 được nhận hoàn toàn nên t-Reassemly tái khởi động cho SN =2.
* Khi t-Reassembly hết hạn (không thể tái hợp các mảnh SN=2 kịp thời), status reporting được kích hoạt. STATUS PDU sẽ bao gồm NACK cho SN = 2.

Trong trường hợp c:

* SN = 3 được nhận một phần. Khi t-Resembly hết hạn (không kịp tái tổ hợp các mảnh có SN = 1 kịp), status report được kích hoạt.
* STATUS PDU sẽ bao gồm NACK cho SN = 1 và SN = 2, và có thể cho cả SN = 3.

GIẢI THÍCH CÁCH SỬ DỤNG CÁC TIMERS CỦA PHẦN TỬ RLC AM



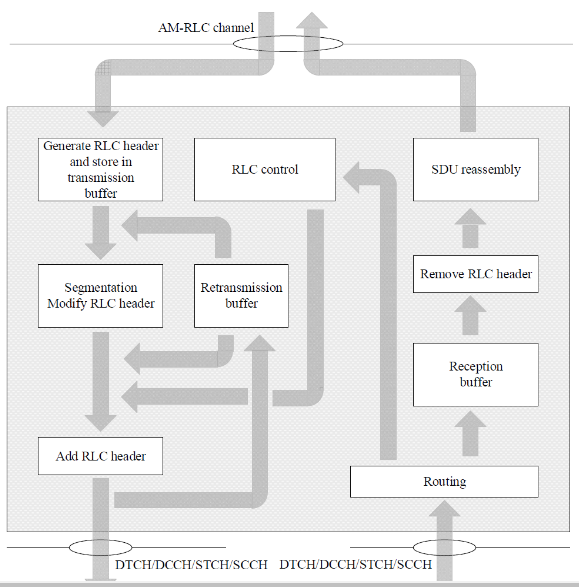
Hình 2. Kích hoạt và gửi STATUS PDU

Một phần tử RLC AM duy trì 3 timers.

* **t-PollRetransmit**: ở phía phát
  + Bắt đầu hoặc tái khởi động khi poll bit được thiết lập ở AMD PDU.
  + Dùng khi STATUS PDU nhận được mô tả ACK/NACK cho POLL\_SN (SN cao nhất được truyền trong lần poll gần nhất).
  + Khi hết hạn, bắt đầu gửi lại dữ liệu và poll.
* **t-Reassembly**: ở phía thu
  + Phát hiện RLC PDUs bị mất.
  + BẮt đầu khi 1 phân mảnh được nhận nhưng các phân mảnh khác của SDU đó đang được chờ nhận.
  + Kết thúc khi SDU được nhận hoàn toàn.
  + Khi hết hạn thì kích hoạt báo cáo trạng thái.
* **t-StatusProhibit**: ở phía thu
  + Ngăn việc truyền thường xuyên của STATUS PDU.
  + Bắt đầu khi 1 STATUS PDU được gửi đi.
  + Hết hạn của t-Reassembly sẽ kích hoạt một báo cáo trạng thái nhưng STATUS PDU chỉ được gửi khi t-StatusProhibit hết hạn.
  + Báo cáo trạng thái được kích hoạt hoặc bới polling (bên phát kích hoạt) hoặc bởi t-Reassembly hết hạn (bên thu kích hoạt).

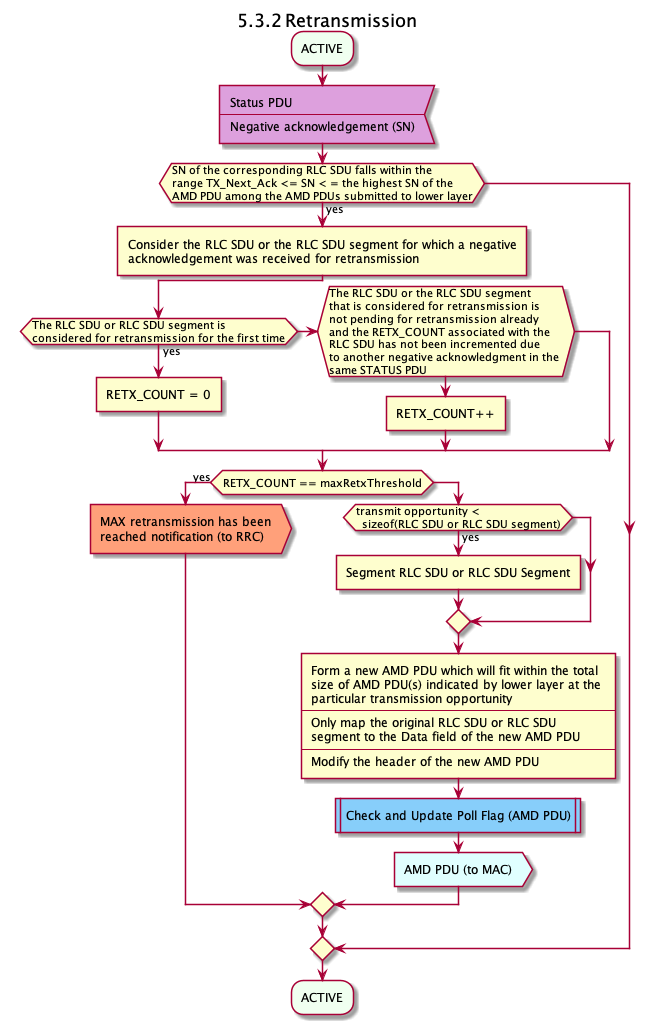
**II. Cách các phần tử AM RLC hoạt động**

* Ở phía phát:
  + RLC SDU được nhận từ lớp trên.
  + RLC chỉ định SN tiếp theo tới SDU. Nó gửi RLC header them vào RC SDU tới lớp MAC.
  + Nếu toàn bộ SDU không thể gửi trong 1 lần truyền, nó sẽ được phân mảnh. Header them vào mảnh lớn nhất có thể được gửi tới MAC.
* Ở phía thu:
  + Phía thu phản hồi thông qua ACK\_SN (positive SDUs) và NACK\_SN (bị mất SDU hoặc bị mất mảnh).
  + Nếu tất cả mảnh của 1 SDU dược nhận chuẩn xác, RLC sẽ chuyển tiếp SDU tới lớp cao hơn.
  + Thứ tự gửi là không quan trọng với RLC.
* Truyền lại ở phía phát:
  + Truyền lại SDU hoặc các mảnh bị mất.
  + Trong khi truyền, RLC có thể phân mảnh lại các phần bị mất của SDU.
  + RLC cho phép một số lần truyền lại tối đa, nếu vượt qua ngưỡng đó, lỗi sẽ được báo cáo lên lớp trên.
* Polling:
  + Thông qua polling, bên phát có thể yêu cầu bên thu gửi trạng thái hiện tại.
  + Bên thu có 1 timer để thường xuyên gửi trạng thái theo chu kỳ.
  + Tương tự, bên phát có 1 timer cho việc gửi polling theo chu kỳ.



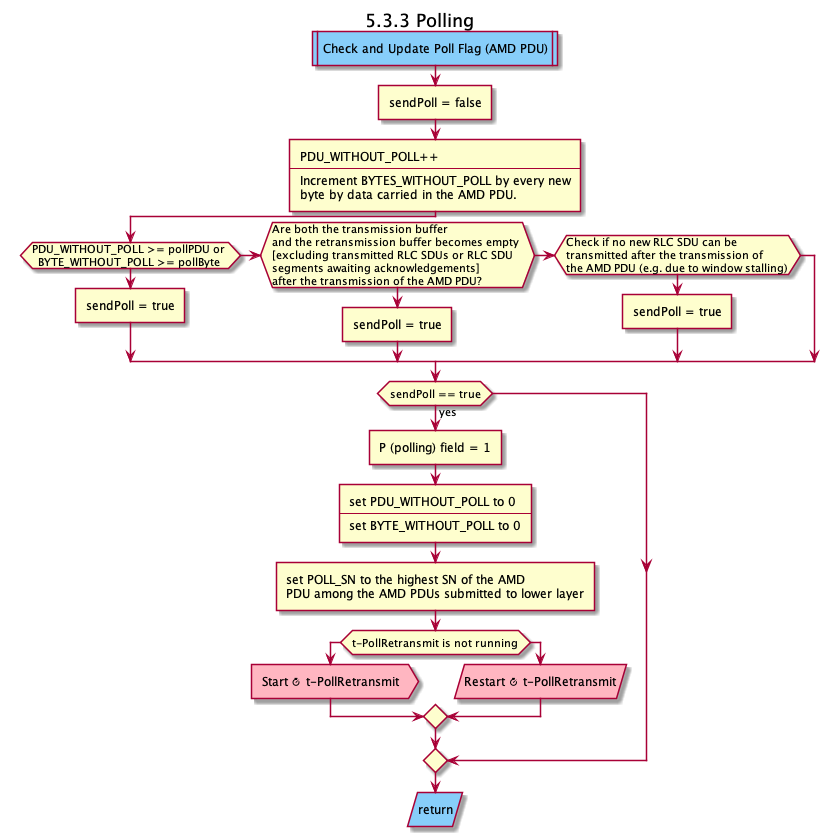
**II. Cơ chế truyền lại của RLC**

1. Lớp RLC nhận một Status PDU mà phản hồi NACK cho RLC SDUs.
2. Lớp RLC sẽ duyệt qua cả positive và negative ack. RLC sẽ đảm bảo rằng SN của RLC SDU tương ứng được phản hồi nằm trong khoảng mong muốn TX\_Next\_Ack <= SN <= giá trị SN cao nhất của AMD PDUs đã được gửi tới lớp thấp hơn.
3. Nếu một negative ack được nhận, bắt đầu đặt biến RETX\_COUNT bằng 0 cho lần truyền lại đầu tiên. RETX\_COUNT sẽ tăng dần cho các lần truyền lại sau đó.
4. Nếu RETX\_COUNT đạt ngưỡng maxRetxThreshold, RLC sẽ được thông báo rằng đã đạt giới hạn truyền lại cho RLC SDU hoặc RLC segment này.
5. Nếu giới hạn RETX\_COUNT chưa đạt giới hạn, RLC sẽ truyền lại RLC SDU hoặc RLC SDU segment.
6. Nếu khả năng truyền lại từ lớp MAC nhỏ hơn kích thước của RLC SDU/ RLC SDU segment, RLC sẽ phân mảnh RLC SDU/ RLC SDU segment.
7. RLC sau đó sẽ định hình AMD PDU để phù hợp với khả năng truyền.
8. RLC sau đó quyết định nếu polling flag trong RLC header cần cập nhật lại.
9. RLC truyền AMD PDU tới lớp MAC.



**III. Polling**

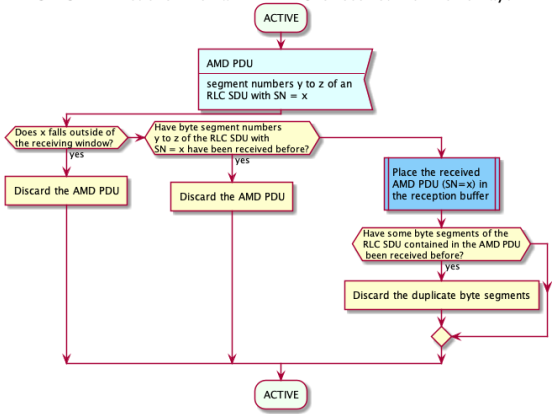
* RLC sử dụng polling flag trong header để thu thập STATUS PDU từ RLC đồng cấp từ phía thu/phát ngược lại. Polling flag được thiết lập nếu:
  + Tổng PDUs được gửi từ lần poll gần nhất chạm tới giá trị pollPDU.
  + Tổng số bytes được truyền từ lần poll gần nhất đạt tới giá trị pollByte.
  + Buffer truyền và truyền lại sẽ trở thành trống sau khi truyền AMD PDU hiện thời.
  + Cửa sổ sẽ ngừng hoạt động sau khi truyền AMD PDU hiện tại. (window stall diễn ra sau khi truyền AMD PDU hiện tại) 🡪 không có RLC SDU mới nào có thể truyền đi.
* Bộ đếm t-PollRetransmit được bắt đầu sau khi một poll được gửi.



**IV. Nhận và tái hợp các gói.**

1. Nhận AMD PDU

* AMD PDU được nhận từ lớp MAC
* RLC sẽ hoạt động nếu RLC PDU SN nằm trong cửa sổ nhận.
  + AM Window size = 2048 nếu SN 12 bit
  + AM Window size = 131072 nếu SN 18 bit
* RLC cũng kiểm tra nếu AMD PDU mang theo RLC SDU segment bytes đã được nhận trước đó hay chưa. RLC sẽ tiếp tục xử lý với các AMD PDU chứa các data bytes mới.
* RLC sau đó thêm AMD PDU vào reception buffer (mô tả ở phần 2).

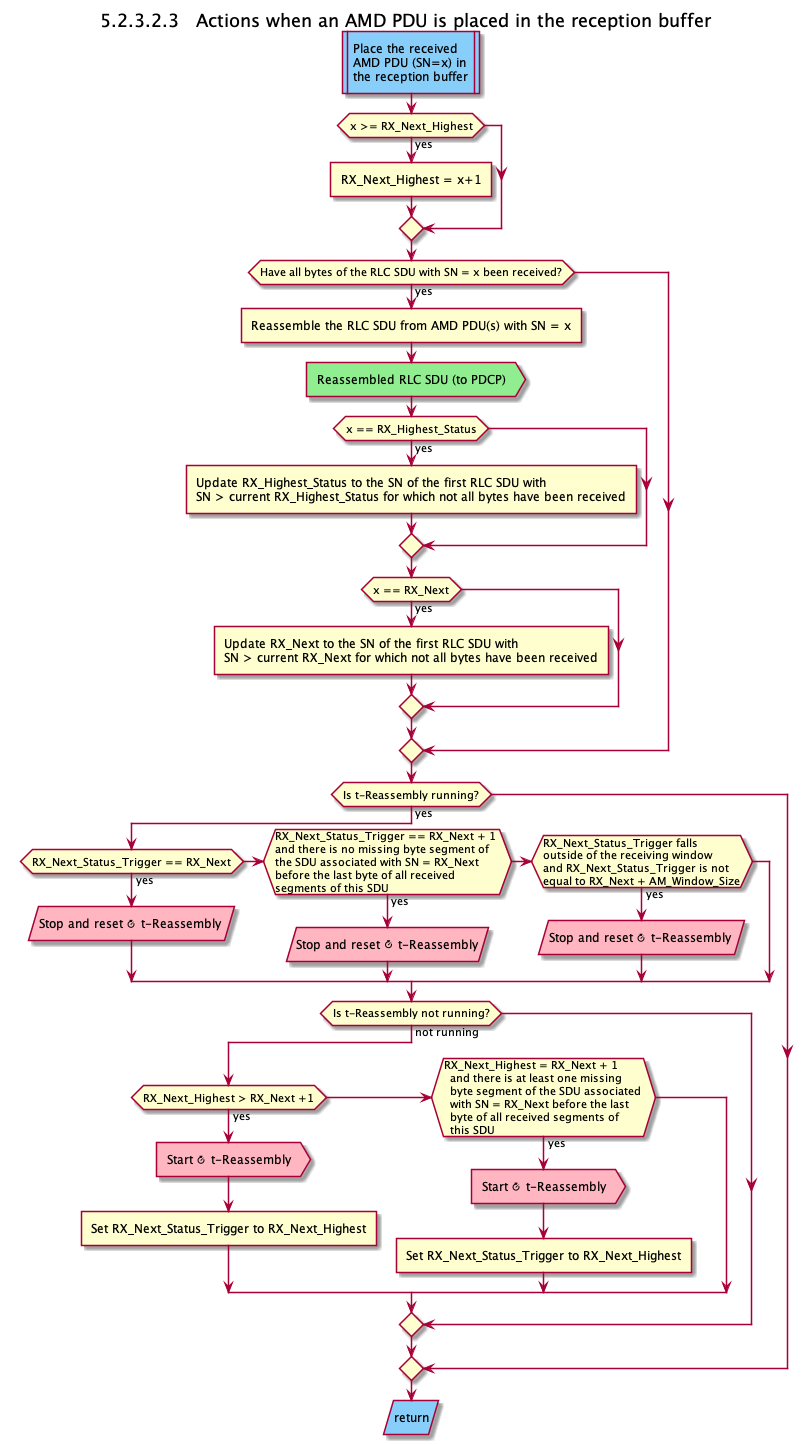


1. **Thêm AMD PDU vào reception buffer**

* Một AMD PDU với SN = x sẽ được lưu vào trong reception buffer
* Nếu SN x nhận được >= RX\_Next\_Highest, RX\_Next\_Highest sẽ tăng lên. Chú ý rằng, RX\_Next\_Highest tương ứng với giá trị ngay sau SN cao nhất đã từng nhận được.
* Khi tất cả các bytes của RLC SDU với SN = x được nhận hoàn toàn, tổ hợp lại các SDU này và báo cáo với lớp trên.
* Nếu x trùng với RX\_Highest\_Status, điều này biểu thị tất cả SNs trước x cũng đã được nhận thành công. RX\_Highest\_Status lúc này được cập nhật với SN của RLC SDU đầu tiên mà có SN > RX\_Highest\_Status hiện tại trong đó, không phải tất cả bytes của SDU này đã được nhận. (Chú ý rằng RX\_Highest\_Status chỉ SN sớm nhất mà tất cả các bytes chưa được nhận hết).
* Nếu x trùng với RX\_Next, RX\_Next được cập nhật với SN của RLC PDU đầu tiên mà có SN > RX\_Next hiện tại trong đó tất cả các byte chưa được nhận hoàn toàn. (segment)
* Nếu t-Reassembly đang chạy, dừng và khởi tạo lại timer này nếu 1 trong số các điều kiện sau thỏa mãn:
  + RX\_Next\_Status\_Trigger = = RX\_Next (đến lúc trigger status)
  + RX\_Next\_Status\_Trigger = = RX\_Next + 1 và không có byte segment nào của SDU đi cùng với SN = RX\_Next bị thiều trước byte cuối cùng của tất cả các mảnh nhận được của SDU này. (SDU có thể bị tách thành các mảnh và phân cho các PDU nhau, đảm bảo các segment này không bị thiếu cho đến khi nhận được byte cuối cùng).
  + RX\_Next\_Status\_Trigger nằm ngoài receiving window (131072) và RX\_Next\_Status\_Trigger != RX\_Next + AM\_Window\_Size (giá trị: 131072).

Chú ý rằng: RX\_Next\_Status\_Trigger là giá trị SN trong quá khứ của PDU gần nhất đã kích hoạt t-Reassembly.

* Nếu t-Reassembly đang không chạy, bắt đầu t-Reassembly và đặt RX\_Next\_Status\_Trigger = RX\_Next\_Highest nếu thỏa mãn 1 trong số các điều kiện:
  + RX\_Next\_Highest > RX\_Next + 1
  + RX\_Next\_Highest = RX\_Next + 1 và có ít nhất 1 byte segment của SDU với SN = Rx\_Next bị mất trước khi byte cuối cùng của SDU này được nhận.



1. **Hoạt động khi t-Reassembly hết hạn**

* Cập nhật RX\_Highest\_Status bằng SN mà Sn này đã từng kích hoạt t-Reassembly trong lần gần nhất và là SN đầu tiên mà tất cả các bytes chưa nhận được. (SN đã kích hoạt t-Reassembly và các bytes thuộc SN này đều chưa nhận được).
* Nếu các thao tác trên dẫn đến RX\_Next\_Status vượt qua giá trị RX\_Highest\_Status + 1, các hoạt động sau sẽ được thực hiện:
  + Bắt đầu t-Reassembly timer
  + Cập nhật RX\_Next\_Status\_Trigger để ghi nhớ giá trị SN mà đã kích hoạt t-Reassembly timer.
* Nếu các hoạt động trên dẫn đến RX\_Next\_Highest tiến tới RX\_Highest\_Status + 1 và có ít nhất 1 byte segment bị mất với SN = RX\_Highest\_Status:
  + Bắt đầu t-Reassembly timer
  + Cập nhật RX\_Next\_Status\_Trigger để ghi nhớ giá trị SN mà đã kích hoạt t-Reassembly timer.

