CHAPTER

13 Retransmission Protocols

Transmissions over wireless channels are subject to errors, for example, due to variations in the received signal quality. To some degree, such variations can be counteracted through link adaptation as will be discussed in Chapter 14. However, receiver noise and unpredictable interference variations cannot be counteracted. Therefore, virtually all wireless communication systems employ some form of *Forward Error Correction* (FEC), adding redundancy to the transmitted signal allowing the receiver to correct errors and tracing its roots to the pioneering work of Shannon [69]. In NR, LDPC coding is used for error correction as discussed in Section 9.2.

Despite the error-correcting code, there will be data units received in error, for example, due to a too high noise or interference level. *Hybrid Automatic Repeat Request* (HARQ), first proposed by Wozencraft and Horstein [72] and relying on a combination of error-correcting coding and retransmission of erroneous data units, is therefore commonly used in many modern communication systems. Data units in error despite the error correcting coding are detected by the receiver, which requests a retransmission from the transmitter.

In NR, three different protocol layers all offer retransmission functionality—MAC, RLC, and PDCP—as already mentioned in the introductory overview in Chapter 6. The reasons for having a multilevel retransmission structure can be found in the trade-off between fast and reliable feedback of the status reports. The hybrid-ARQ mechanism in the MAC layer targets very fast retransmissions and, consequently, feedback on success or failure of the downlink transmission is provided to the gNB after each received transport block (for uplink transmission no explicit feedback needs to be transmitted as the receiver and scheduler are in the same node). Although it is in principle possible to attain a very low error probability of the hybrid-ARQ feedback, it comes at a cost in transmission resources such as power. In many cases, a feedback error rate of 0.1−1% is reasonable, which results in a hybrid-ARQ residual error rate of a similar order. In many cases this residual error rate is sufficiently low, but there are cases when this is not the case. One obvious case is services requiring ultra-reliable delivery of data combined with low latency. In such cases, either the feedback error rate needs to be decreased and the increased cost in feedback signaling has to be accepted, or additional retransmissions can be performed without relying on feedback signaling, which comes at a decreased spectral efficiency.

5G NR: The Next Generation Wireless Access Technology. DOI: https://doi.org/10.1016/B978-0-12-814323-0.00013-2

A low error rate is not only of interest for URLLC type of services, but is also important from a data-rate perspective. High data rates with TCP may require virtually error-free delivery of packets to the TCP layer. As an example, for sustainable data rates exceeding 100 Mbit/s, a packet-loss probability less than 10^{-5} is required [65]. The reason is that TCP assumes packet errors to be due to congestion in the network. Any packet error therefore triggers the TCP congestion-avoidance mechanism with a corresponding decrease in data rate.

Compared to the hybrid-ARQ acknowledgments, the RLC status reports are transmitted relatively infrequently and thus the cost of obtaining a reliability of 10^{-5} or lower is relatively small. Hence, the combination of hybrid-ARQ and RLC attains a good combination of small round-trip time and a modest feedback overhead where the two components complement each other—fast retransmissions due to the hybrid-ARQ mechanism and reliable packet delivery due to the RLC.

The PDCP protocol is also capable of handling retransmissions, as well as ensuring in-sequence delivery. PDCP-level retransmissions are mainly used in the case of inter-gNB handover as the lower protocols in this case are flushed. Not-yet-acknowledged PDCP PDUs can be forwarded to the new gNB and transmitted to the device. In the case that some of these were already received by the device, the PDCP duplicate detection mechanism will discard the duplicates. The PDCP protocol can also be used to obtain selection diversity by transmitting the same PDUs on multiple carriers. The PDCP in the receiving end will in this case remove any duplicates in case the same information was received successfully on multiple carriers.

In the following sections, the principles behind the hybrid-ARQ, RLC, and PDCP protocols will be discussed in more detail. Note that these protocols are present also in LTE where they to a large extent provide the same functionality. However, the NR versions are enhanced to significantly reduce the delays.

13.1 HYBRID-ARQ WITH SOFT COMBINING

The hybrid-ARQ protocol is the primary way of handling retransmissions in NR. In case of an erroneously received packet, a retransmission is requested. However, despite it not being possible to decode the packet, the received signal still contains information, which is lost by discarding erroneously received packets. This shortcoming is addressed by *hybrid-ARQ with soft combining*. In hybrid-ARQ with soft combining, the erroneously received packet is stored in a buffer memory and later combined with the retransmission to obtain a single, combined packet that is more reliable than its constituents. Decoding of the error-correction code operates on the combined signal.

Although the protocol itself primarily resides in the MAC layer, there is also physical layer functionality involved in the form of soft combining.

Retransmissions of codeblock groups, that is, retransmission of a part of the transport block, are handled by the physical layer from a specification perspective, although it could equally well have been described as part of the MAC layer.

The basis for the NR hybrid-ARQ mechanism is, similarly to LTE, a structure with multiple stop-and-wait protocols, each operating on a single transport block. In a stop-and-wait protocol, the transmitter stops and waits for an acknowledgment after each transmitted transport block. This is a simple scheme; the only feedback required is a single bit indicating positive or negative acknowledgment of the transport block. However, since the transmitter stops after each transmission, the throughput is also low. Therefore, multiple stop-and-wait processes operating in parallel are used such that, while waiting for acknowledgment from one process, the transmitter can transmit data to another hybrid-ARQ process. This is illustrated in Fig. 13.1; while processing the data received in the first hybrid-ARQ process the receiver can continue to receive using the second process, etc. This structure, multiple hybrid-ARQ processes operating in parallel to form one hybrid-ARQ entity, combines the simplicity of a stop-and-wait protocol while still allowing continuous transmission of data, and is used in LTE as well as NR.

There is one hybrid-ARQ entity per carrier the receiver is connected to. Spatial multiplexing of more than four layers to a single device in the downlink, where two transport blocks can be transmitted in parallel on the same transport channel as described in Section 9.1, is supported by one hybrid-ARQ entity having two sets of hybrid-ARQ processes with independent hybrid-ARQ acknowledgments.

NR uses an *asynchronous* hybrid-ARQ protocol in both downlink and uplink, that is, the hybrid-ARQ process which the downlink or uplink transmission relates to is explicitly signaled as part of the downlink control information (DCI). LTE uses the same scheme for the downlink but not for the uplink, where LTE uses a synchronous protocol (although later LTE releases added support for an

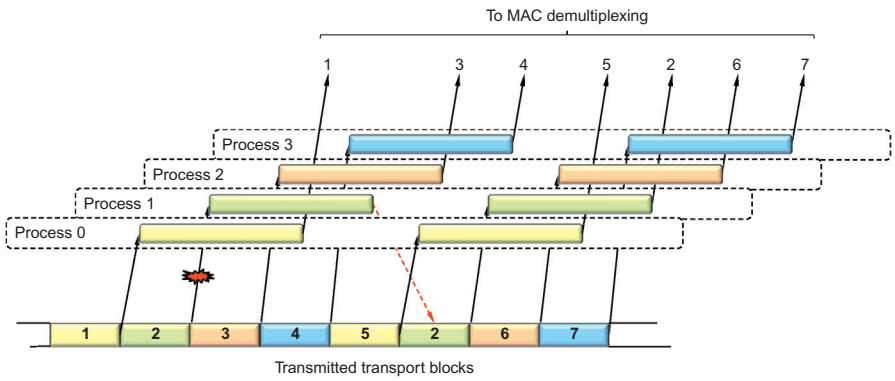

FIGURE 13.1

Multiple hybrid-ARQ processes.

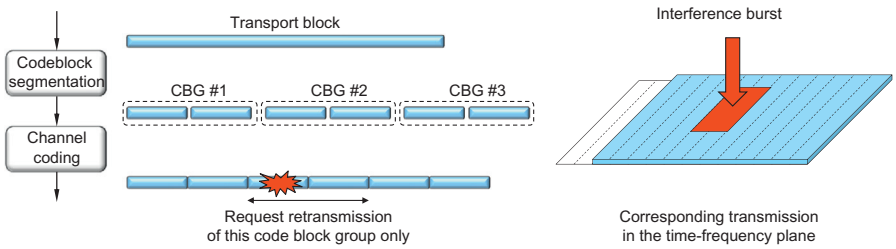

FIGURE 13.2

Codeblock-group retransmission.

asynchronous protocol as well). There are several reasons why NR adopted an asynchronous protocol in both directions. One reason is that synchronous hybrid-ARQ operation does not allow dynamic TDD. Another reason is that operation in unlicensed spectra, to be introduced in later NR releases, is more efficient with asynchronous operation as it cannot be guaranteed that the radio resources are available at the time for a synchronous retransmission. Thus, NR settled for an asynchronous scheme in both uplink and downlink with up to 16 processes. Having a larger maximum number of hybrid-ARQ processes than in LTE[1] is motivated by the possibility for remote radio heads, which incurs a certain front-haul delay, together with the shorter slot durations at high frequencies. It is important though, that the larger number of maximum hybrid-ARQ processes does not imply a longer roundtrip time as not all processes need to be used, it is only an upper limit of the number of processes possible to address.

Large transport block sizes are segmented into multiple codeblocks prior to coding, each with its own 24-bit CRC (in addition to the overall transport-block CRC). This was discussed already in Section 9.2 and the reason is primarily complexity; the size of a codeblock is large enough to give good performance while still having a reasonable decoding complexity. Since each codeblock has its own CRC, errors can be detected on individual codeblocks as well as on the overall transport block. A relevant question is if retransmission should be limited to transport blocks or whether there are benefits of retransmitting only the codeblocks that are erroneously received. For the very large transport block sizes used to support data rates of several gigabits per second, there can be hundreds of codeblocks in a transport block. If only one or a few of them are in error, retransmitting the whole transport block results in a low spectral efficiency compared to retransmitting only the erroneous codeblocks. One example where only some codeblocks are in error is a situation with bursty interference where some OFDM symbols are hit more severely than others, as illustrated in Fig. 13.2, for example, due to one downlink transmission preempting another as discussed in Section 14.1.2.

[1]In LTE, eight processes are used for FDD and up to 15 processes for TDD, depending on the uplink–downlink configuration.

To correctly receive the transport block for the example above, it is sufficient to retransmit the erroneous codeblocks. At the same time, the control signaling overhead would be too large if individual codeblocks can be addressed by the hybrid-ARQ mechanism. Therefore, so-called *codeblock groups* (CBGs) are defined. If per-CBG retransmission is configured, feedback is provided per CBG instead of per transport block and only the erroneously received codeblock groups are retransmitted, which consumes less resources than retransmitting the whole transport block. Two, four, six, or eight codeblock groups can be configured with the number of codeblocks per codeblock group varying as a function of the total number of codeblocks in the initial transmission. Note that the codeblock group a codeblock belongs to is determined from the initial transmission and does not change between the transmission attempts. This is to avoid error cases which could arise if the codeblocks were repartitioned between two retransmissions.

The CBG retransmissions are handled as part of the physical layer from a specification perspective. There is no fundamental technical reason for this but rather a way to reduce the specification impact from CBG-level retransmissions. A consequence of this is that it is not possible, in the same hybrid-ARQ process, to mix transmission of new CBGs belonging to another transport block with retransmissions of CBGs belonging to the incorrectly received transport block.

13.1.1 SOFT COMBINING

An important part of the hybrid-ARQ mechanism is the use of *soft combining*, which implies that the receiver combines the received signal from multiple transmission attempts. By definition, a hybrid-ARQ retransmission must represent the same set of information bits as the original transmission. However, the set of coded bits transmitted in each retransmission may be selected differently as long as they represent the same set of information bits. Depending on whether the retransmitted bits are required to be identical to the original transmission or not, the soft combining scheme is often referred to as *Chase combining*, first proposed in Ref. [22], or *Incremental Redundancy* (IR), which is used in NR. With incremental redundancy, each retransmission does not have to be identical to the original transmission. Instead, *multiple sets* of coded bits are generated, each representing the same set of information bits [67,71]. The rate-matching functionality of NR, described in Section 9.3, is used to generate different sets of coded bits as a function of the redundancy version as illustrated in Fig. 13.3.

In addition to a gain in accumulated received E_b/N_0, incremental redundancy also results in a coding gain for each retransmission (until the mother code rate is reached). The gain with incremental redundancy compared to pure energy accumulation (Chase combining) is larger for high initial code rates [24]. Furthermore, as shown in Ref. [33], the performance gain of incremental redundancy compared to Chase combining can also depend on the relative power difference between the transmission attempts.

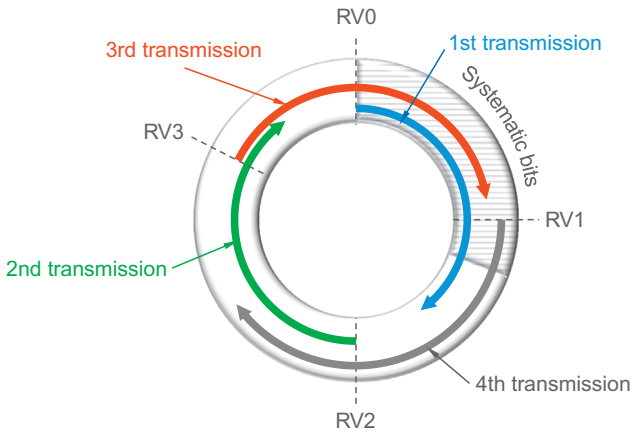

FIGURE 13.3

Example of incremental redundancy.

In the discussion so far, it has been assumed that the receiver has received all the previously transmitted redundancy versions. If all redundancy versions provide the same amount of information about the data packet, the order of the redundancy versions is not critical. However, for some code structures, not all redundancy versions are of equal importance. This is the case for the LDPC codes used in NR; the systematic bits are of higher importance than the parity bits. Hence, the initial transmission should at least include all the systematic bits and some parity bits. In the retransmission(s), parity bits not in the initial transmission can be included. This is the background to why systematic bits are inserted first in the circular buffer in Section 9.3. The starting points in the circular buffer are defined such that both RV0 and RV3 are self-decodable, that is, includes the systematic bits under typical scenarios. This is also the reason RV3 is located after nine o'clock in Fig. 13.3, as this allows more of the systematic bits to be included in the transmission. With the default order of the redundancy versions 0, 2, 3, 1, every second retransmission is typically self-decodable.

Hybrid ARQ with soft combining, regardless of whether Chase or incremental redundancy is used, leads to an implicit reduction of the data rate by means of retransmissions and can thus be seen as implicit link adaptation. However, in contrast to link adaptation based on explicit estimates of the instantaneous channel conditions, hybrid-ARQ with soft combining implicitly adjusts the coding rate based on the result of the decoding. In terms of overall throughput this kind of implicit link adaptation can be superior to explicit link adaptation, as additional redundancy is only added *when needed*—that is, when previous higher-rate transmissions were not possible to decode correctly. Furthermore, as it does not try to predict any channel variations, it works equally well, regardless of the speed at which the terminal is moving. Since implicit link adaptation can provide a gain in

system throughput, a valid question is why explicit link adaptation is necessary at all. One major reason for having explicit link adaptation is the reduced delay. Although relying on implicit link adaptation alone is sufficient from a system throughput perspective, the end-user service quality may not be acceptable from a delay perspective.

For proper operation of soft combining, the receiver needs to know when to perform soft combining prior to decoding and when to clear the soft buffer—that is, the receiver needs to differentiate between the reception of an initial transmission (prior to which the soft buffer should be cleared) and the reception of a retransmission. Similarly, the transmitter must know whether to retransmit erroneously received data or to transmit new data. This is handled by the *new-data indicator* as discussed further below for downlink and uplink hybrid-ARQ, respectively.

13.1.2 DOWNLINK HYBRID-ARQ

In the downlink, retransmissions are scheduled in the same way as new data—that is, they may occur at any time and at an arbitrary frequency location within the downlink cell bandwidth. The scheduling assignment contains the necessary hybrid-ARQ-related control signaling—hybrid-ARQ process number, new-data indicator, CBGTI, and CBGFI in case per-CBG retransmission is configured, as well as information to handle the transmission of the acknowledgment in the uplink such as timing and resource indication information.

Upon receiving a scheduling assignment in the DCI, the receiver tries to decode the transport block, possibly after soft combining with previous attempts as described above. Since transmissions and retransmissions are scheduled using the same framework in general, the device needs to know whether the transmission is a new transmission, in which case the soft buffer should be flushed, or a retransmission, in which case soft combining should be performed. Therefore, an explicit *new-data indicator* is included for the scheduled transport block as part of the scheduling information transmitted in the downlink. The new-data indicator is toggled for a new transport block—that is, it is essentially a single-bit sequence number. Upon reception of a downlink scheduling assignment, the device checks the new-data indicator to determine whether the current transmission should be soft combined with the received data currently in the soft buffer for the hybrid-ARQ process in question, or if the soft buffer should be cleared.

The new-data indicator operates on the transport-block level. However, if per-CBG retransmissions are configured, the device needs to know which CBGs are retransmitted and whether the corresponding soft buffer should be flushed or not. This is handled through two additional information fields present in the DCI in case per-CBG retransmission is configured, the *CBG Transmit Indicator* (CBGTI) and the *CBG Flush Indicator* (CBGFI). The CBGTI is a bitmap indicating whether a certain CBG is present in the downlink transmission or not (see

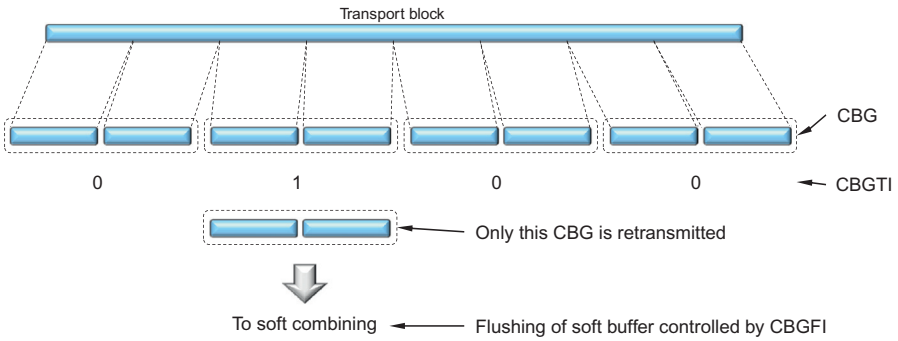

FIGURE 13.4

Illustration of per-CBG retransmission.

Fig. 13.4). The CBGFI is a single bit, indicating whether the CBGs indicated by the CBGTI should be flushed or whether soft combining should be performed.

The result of the decoding operation—a positive acknowledgment in the case of a successful decoding and a negative acknowledgment in the case of unsuccessful decoding—is fed back to the gNB as part of the uplink control information. If CBG retransmissions are configured, a bitmap with one bit per CBG is fed back instead of a single bit representing the whole transport block.

13.1.3 UPLINK HYBRID-ARQ

The uplink uses the same asynchronous hybrid-ARQ protocol as the downlink. The necessary hybrid-ARQ-related information—hybrid-ARQ process number, new-data indicator, and, if per-CBG retransmission is configured, the CBGTI—is included in the scheduling grant.

To differentiate between new transmissions and retransmissions of data, the new-data indicator is used. Toggling the new-data indicator requests transmission of a new transport block, otherwise the previous transport block for this hybrid-ARQ process should be retransmitted (in which case the gNB can perform soft combining). The CBGTI is used in a similar way as in the downlink, namely to indicate the codeblock groups to retransmit in the case of per-CBG retransmission. Note that no CBGFI is needed in the uplink as the soft buffer is located in the gNB which can decide whether to flush the buffer or not based on the scheduling decisions.

13.1.4 TIMING OF UPLINK ACKNOWLEDGMENTS

In LTE, the time from downlink data reception to transmission of the acknowledgment is fixed in the specifications. This is possible for full-duplex transmission, for example, FDD, in which case the acknowledgment is transmitted almost

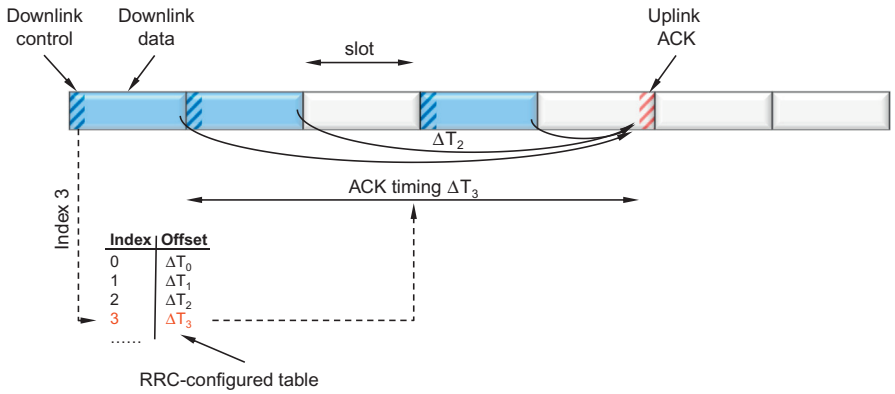

FIGURE 13.5

Determining the acknowledgment timing.

3 ms after the end of data reception in LTE.[2] A similar approach can be used if the uplink−downlink allocation is semistatically configured in the case of half-duplex operation, for example, semistatic TDD as in LTE. Unfortunately, this type of scheme with predefined timing instants for the acknowledgments does not blend well with dynamic TDD, one of the cornerstones of NR, as an uplink opportunity cannot be guaranteed a fixed time after the downlink transmission due to the uplink−downlink direction being dynamically controlled by the scheduler. Coexistence with other TDD deployments in the same frequency band may also impose restrictions when it is desirable, or possible, to transmit in the uplink. Furthermore, even if it would be possible, it may not be desirable to change the transmission direction from downlink to uplink in each slot as this would increase the switching overhead. Consequently, a more flexible scheme capable of dynamically controlling when the acknowledgment is transmitted is adopted in NR.

The hybrid-ARQ timing field in the downlink DCI is used to control the transmission timing of the acknowledgment in the uplink. This three-bit field is used as an index into an RRC-configured table providing information on when the hybrid-ARQ acknowledgment should be transmitted relative to the reception of the PDSCH (see Fig. 13.5). In this particular example, three slots are scheduled in the downlink before an acknowledgment is transmitted in the uplink. In each downlink assignment, different acknowledgment timing indices have been used, which in combination with the RRC-configured table result in all three slots being acknowledged at the same time (multiplexing of these acknowledgments in the same slot is discussed below).

[2]The time depends on the timing advance value. For the largest possible timing advance, the time is 2.3 ms in LTE.

Table 13.1 Minimum Processing Time (PDSCH Mapping Type A, Feedback on PUCCH)

DM-RS Configuration	Device Capability	Subcarrier Spacing				LTE Rel 8
		15 kHz	30 kHz	60 kHz	120 kHz	
Front-loaded	Baseline	0.57 ms	0.36 ms	0.30 ms	0.18 ms	2.3 ms
	Aggressive	0.18–0.29 ms	0.08–0.17 ms			
Additional	Baseline	0.92 ms	0.46 ms	0.36 ms	0.21 ms	
	Aggressive	0.85 ms	0.4 ms			

Furthermore, NR is designed with very low latency in mind and is therefore capable of transmitting the acknowledgment much sooner after the end of the downlink data reception than the corresponding LTE timing relation. All devices support the baseline processing times listed in Table 13.1, with even faster processing optionally supported by some devices. The capability is reported per subcarrier spacing. One part of the processing time is constant in symbols across different subcarrier spacing, that is, the time in microseconds scales with the subcarrier spacing, but there is also a part of the processing time fixed in microseconds and independent of the subcarrier spacing. Hence, the processing times listed in the table are not directly proportional to the subcarrier spacing although there is a dependency. There is also a dependency on the reference signal configuration; if the device is configured with additional reference signal occasions later in the slot, the device cannot start the processing until at least some of these reference signals have been received and the overall processing time is longer. Nevertheless, the processing is much faster than the corresponding LTE case as a result of stressing the importance of low latency in the NR design.

For proper transmission of the acknowledgment it is not sufficient for the device to know *when* to transmit, which is obtained from the timing field discussed above, but also *where* in the resource domain (frequency resources and, for some PUCCH formats, the code domain). In the original LTE design, this is primarily obtained from the location of the PDCCH scheduling the transmission. For NR with its flexibility in the transmission timing of the acknowledgment, such a scheme is not sufficient. In the case that two devices are instructed to transmit their acknowledgment at the same time even if they were scheduled at different time instants, it is necessary to provide the devices with separate resources. This is handled through the *PUCCH resource indicator*, which is a three-bit index selecting one of eight RRC-configured resource sets as described in Section 10.2.7.

13.1.5 MULTIPLEXING OF HYBRID-ARQ ACKNOWLEDGMENTS

In the previous section, the timing of the hybrid-ARQ acknowledgments in the example was such that multiple transport blocks need to be acknowledged at the

same time. Other examples where multiple acknowledgments need to be transmitted in the uplink at the same time are carrier aggregation and per-CBG retransmissions. NR therefore supports multiplexing of acknowledgments for multiple transport blocks received by a device into one multi-bit acknowledgment message. The multiple bits can be multiplexed using either a semi-static codebook or a dynamic codebook with RRC configuration selecting between the two.

The semi-static codebook can be viewed as a matrix consisting of a time-domain dimension and a component-carrier (or CBG or MIMO layer) dimension, both of which are semi-statically configured. The size in the time domain is given by the maximum and minimum hybrid-ARQ acknowledgment timings configured in Table 13.1, and the size in the carrier domain is given by the number of simultaneous transport blocks (or CBGs) across all component carriers. An example is provided in Fig. 13.6, where the acknowledgment timings are one, two, three, and four, respectively, and three carriers, one with two transport blocks, one with one transport block, and one with four CBGs, are configured. Since the codebook size is fixed, the number of bits to transmit in a hybrid-ARQ report is known ($4 \cdot 7 = 28$ bits in the example in Fig. 13.6) and the appropriate format for the uplink control signaling can be selected. Each entry in the matrix represents the

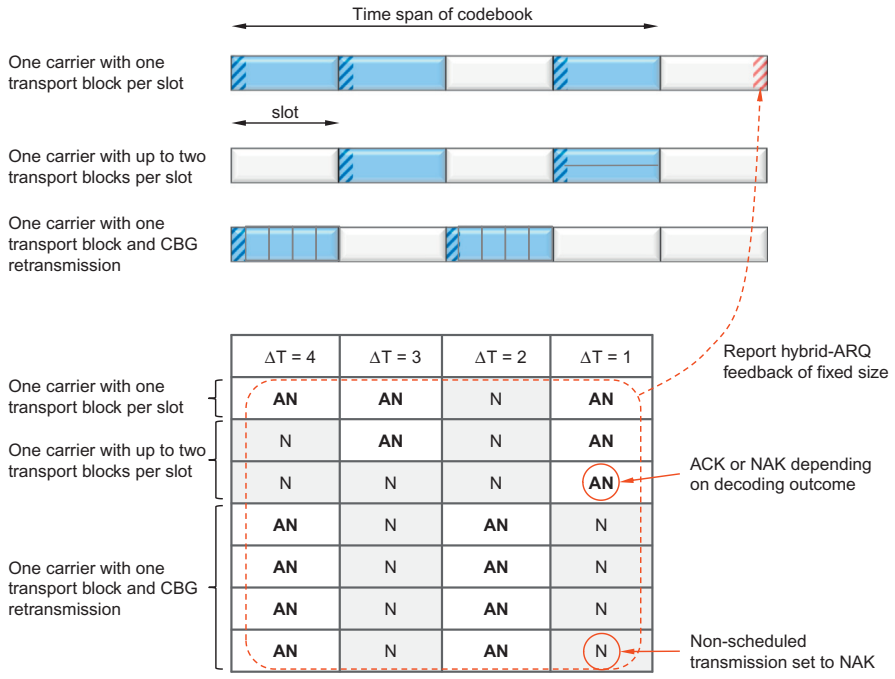

FIGURE 13.6

Example of semistatic hybrid-ARQ acknowledgment codebook.

decoding outcome, positive or negative acknowledgment, of the corresponding transmission. Not all transmission opportunities possible with the codebook are used in this example and for entries in the matrix without a corresponding transmission, a negative acknowledgment is transmitted. This provides robustness; in the case of missed downlink assignment a negative acknowledgment is provided to the gNB, which can retransmit the missing transport block (or CBG).

One drawback with the semistatic codebook is the potentially large size of a hybrid-ARQ report. For a small number of component carriers and no CBG retransmissions, this is less of a problem, but if a large number of carriers and codeblock groups are configured out of which only a small number is simultaneously used, this may become more of an issue.

To address the drawback of a potentially large semi-static codebook size in some scenarios, NR also supports a dynamic codebook. In fact, this is the default codebook used unless the system is configured otherwise. With a dynamic codebook, only the acknowledgment information for the *scheduled* carriers[3] is included in the report, instead of all carriers, scheduled or not, as is the case with a semi-static codebook. Hence, the size of the codebook (the matrix in Fig. 13.6) is dynamically varying as a function of the number of scheduled carriers. In essence, only the bold entries in the example in Fig. 13.6 would be included in the hybrid-ARQ report and the non-bold entries with a gray background (which correspond to non-scheduled carriers) would be omitted. This reduces the size of the acknowledgment message.

A dynamic codebook would be straightforward if there were no errors in the downlink control signaling. However, in the presence of an error in the downlink control signaling, the device and gNB may have different understanding on the number of scheduled carriers, which would lead to an incorrect codebook size and possibly corrupt the feedback report for all carriers, and not only for the ones for which the downlink controls signaling was missed. Assume, as an example, that the device was scheduled for downlink transmission in two subsequent slots but missed the PDCCH and hence scheduling assignment for the first slot. In response the device will transmit an acknowledgment for the second slot only, while the gNB tries to receive acknowledgments for two slots, leading to a mismatch.

To handle these error cases, NR uses the *downlink assignment index* (DAI) included in the DCI containing the downlink assignment. The DAI field is further split into two parts, a counter DAI (cDAI) and, in the case of carrier aggregation, a total DAI (tDAI). The counter DAI included in the DCI indicates the number of scheduled downlink transmissions up to the point the DCI was received in a carrier first, time second manner. The total DAI included in the DCI indicates the total number of downlink transmissions across all carriers up to this point in time,

[3]The description here uses the term "carrier" but the same principle is equally applicable to per-CBG retransmission or multiple transport blocks in the case of MIMO and "transmission instant" is a more generic term, albeit the description would be harder to read.

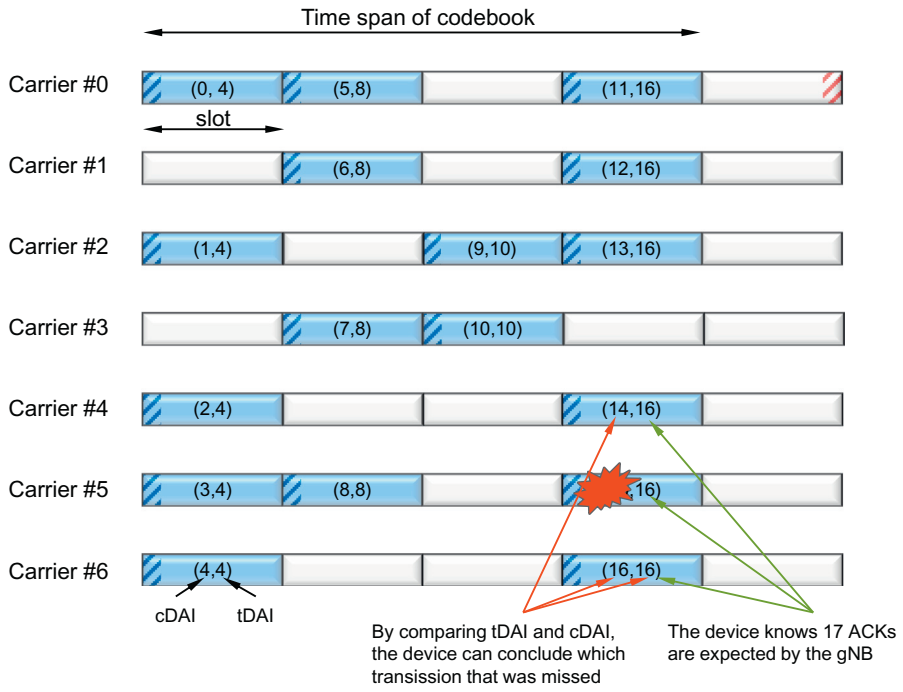

FIGURE 13.7

Example of dynamic hybrid-ARQ acknowledgment codebook.

that is, the highest cDAI at the current point in time (see Fig. 13.7 for an example). The counter DAI and total DAI are represented with decimal numbers with no limitation; in practice two bits are used for each and the numbering will wrap around, that is, what is signaled is the numbers in the figure modulo four. As seen in this example, the dynamic codebook needs to account for 17 acknowledgments (numbered 0−16). This can be compared with the semistatic codebook which would require 28 entries regardless of the number of transmissions.

Furthermore, in this example, one transmission on component carrier five is lost. Without the DAI mechanism, this would result in misaligned codebooks between the device and the gNB. However, as long as the device receives at least one component carrier, it knows the value of the total DAI and hence the size of the codebook at this point in time. Furthermore, by checking the values received for the counter DAI, it can conclude which component carrier was missed and that a negative acknowledgment should be assumed in the codebook for this position.

In the case that CBG retransmission is configured for some of the carriers, the dynamic codebook is split into two parts, one for the non-CBG carriers and one for the CBG carriers. Each codebook is handled according to the principles

outlined above. The reason for the split is that for the CBG carriers, the device needs to generate feedback for each of these carriers according to the largest CBG configuration.

13.2 RLC

The *radio-link control* (RLC) protocol takes data in the form of RLC SDUs from PDCP and delivers them to the corresponding RLC entity in the receiver by using functionality in MAC and physical layers. The relation between RLC and MAC, including multiplexing of multiple logical channels into a single transport channel, is illustrated in Fig. 13.8.

There is one RLC entity per logical channel configured for a device with the RLC entity being responsible for one or more of:

- Segmentation of RLC SDUs;
- Duplicate removal; and
- RLC retransmission.

Unlike LTE, there is no support for concatenation or in-sequence delivery in the RLC protocol. This is a deliberate choice done to reduce the overall latency as discussed further in the following sections. It has also impacted the header design. Also, note that the fact that there is one RLC entity per logical channel and one hybrid-ARQ entity per cell (component carrier) implies that RLC retransmissions can occur on a different cell (component carrier) than the original transmission. This is not the case for the hybrid-ARQ protocol where retransmissions are bound to the same component carrier as the original transmission.

Different services have different requirements; for some services (for example, transfer of a large file), error-free delivery of data is important, whereas for other applications (for example, streaming services), a small amount of missing packets is not a problem. The RLC can therefore operate in three different modes, depending on the requirements from the application:

- *Transparent mode* (TM), where the RLC is completely transparent and is essentially bypassed. No retransmissions, no duplicate detection, and no segmentation/reassembly take place. This configuration is used for control-plane broadcast channels such as BCCH, CCCH, and PCCH, where the information should reach multiple users. The size of these messages is selected such that all intended devices are reached with a high probability and hence there is neither need for segmentation to handle varying channel conditions, nor retransmissions to provide error-free data transmission. Furthermore, retransmissions are not feasible for these channels as there is no possibility for the device to feedback status reports as no uplink has been established.

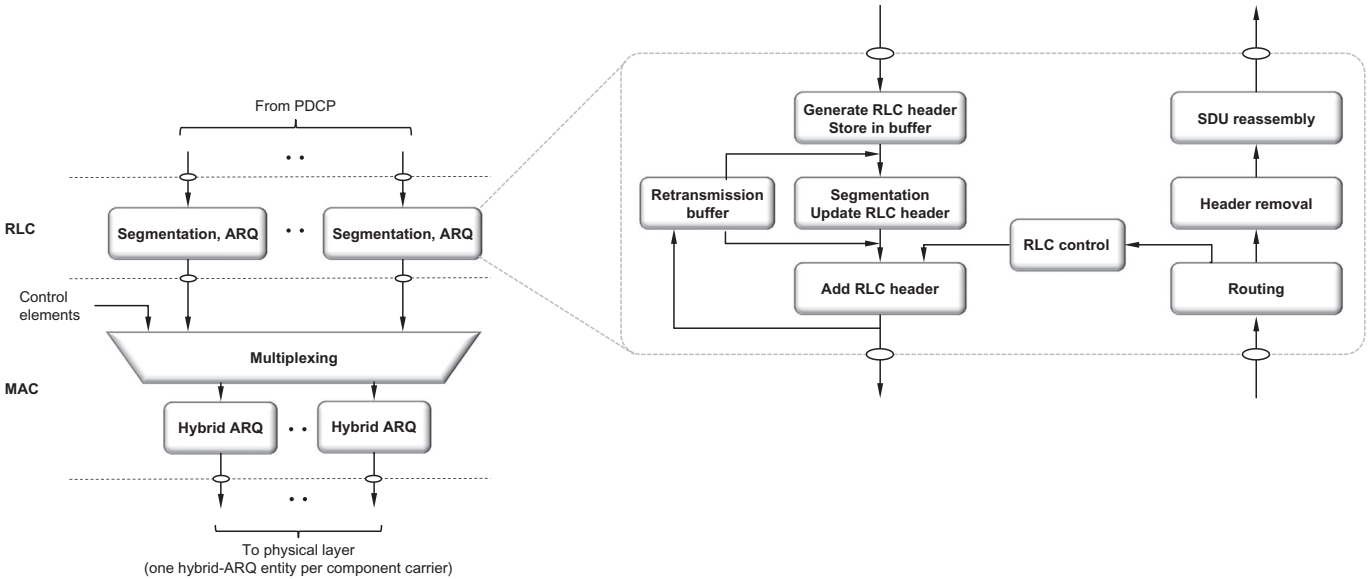

FIGURE 13.8

MAC and RLC.

- *Unacknowledged mode* (UM) supports segmentation but not retransmissions. This mode is used when error-free delivery is not required, for example, voice-over-IP.
- *Acknowledged mode* (AM) is the main mode of operation for the DL-SCH and UL-SCH. Segmentation, duplicate removal, and retransmissions of erroneous data are all supported.

In the following sections, the operation of the RLC protocol is described, focusing on acknowledged mode.

13.2.1 SEQUENCE NUMBERING AND SEGMENTATION

In unacknowledged and acknowledged modes, a sequence number is attached to each incoming SDU using 6 or 12 bits for unacknowledged mode and 12 or 18 bits for acknowledged mode. The sequence number is included in the RLC PDU header in Fig. 13.9. In the case of a non-segmented SDU, the operation is straightforward; the RLC PDU is simply the RLC SDU with a header attached. Note that this allows the RLC PDUs to be generated in advance as the header, in the absence of segmentation, does not depend on the scheduled transport block size. This is beneficial from a latency perspective and the reason the header structure is changed compared to the one used in LTE.

However, depending on the transport-block size after MAC multiplexing, the size of (the last) of the RLC PDUs in a transport block may not match the RLC SDU size. To handle this, an SDU can be segmented into multiple segments. If no segmentation takes place, padding would need to be used instead, leading to degraded spectral efficiency. Hence, dynamically varying the number of RLC PDUs used to fill the transport block, together with segmentation to adjust the size of the last RLC PDU, ensures the transport block is efficiently utilized.

Segmentation is simple; the last preprocessed RLC SDU can be split into two segments, the header of the first segment is updated, and to the second segment a new header is added (which is not time critical as it is not being transmitted in the current transport block). Each SDU segment carries the same sequence number as the original unsegmented SDU and this sequence number is part of the RLC header. To distinguish whether the PDU contains a complete SDU or a segment, a *segmentation information* (SI) field is also part of the RLC header, indicating whether the PDU is a complete SDU, the first segment of the SDU, the last segment of the SDU, or a segment between the first and last segments of the SDU. Furthermore, in the case of a segmented SDU, a 16-bit *segmentation offset* (SO) is included in all segments except the first one and used to indicate which byte of the SDU the segment represents. There is also a *poll bit* (P) in the header used to request a status report for acknowledged mode as described further below, and a *data/control indicator*, indicating whether the RLC PDU contains data to/from a logical channel or control information required for RLC operation.

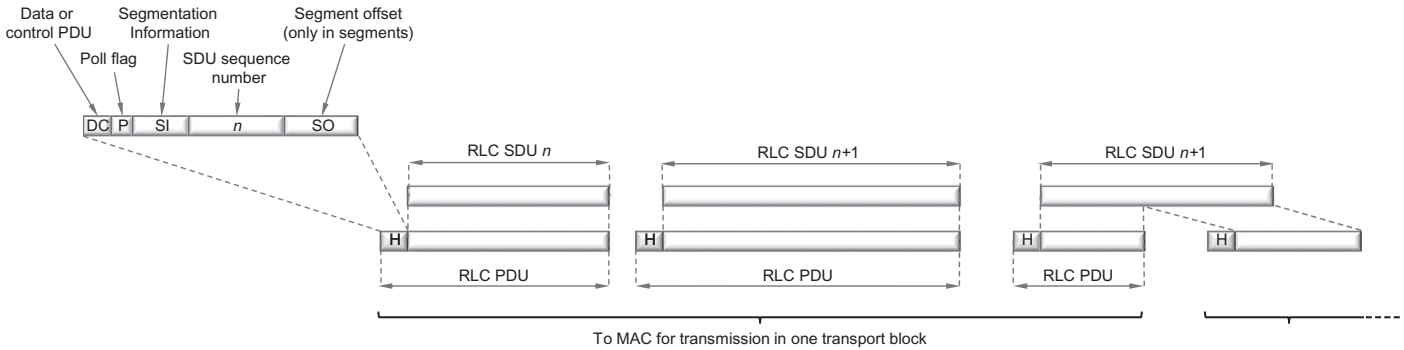

FIGURE 13.9

Generation of RLC PDUs from RLC SDUs (acknowledged mode assumed for the header structure).

The header structure above holds for acknowledged mode. The header for unacknowledged mode is similar but does not include either the poll bit or the data/control indicator. Furthermore, the sequence number is included in the case of segmentation only.

In LTE, the RLC can also perform concatenation of RLC SDUs into a single PDU. However, this functionality is not present in NR in order to reduce latency. If concatenation would be supported, an RLC PDU cannot be assembled until the uplink grant is received as the scheduled transport-block size is not known in advance. Consequently, the uplink grant must be received well in advance to allow sufficient processing time in the device. Without concatenation, the RLC PDUs can be assembled in advance, prior to receiving the uplink grant, and thereby reducing the processing time required between receiving an uplink grant and the actual uplink transmission.

13.2.2 ACKNOWLEDGED MODE AND RLC RETRANSMISSIONS

Retransmission of missing PDUs is one of the main functionalities of the RLC in acknowledged mode. Although most of the errors can be handled by the hybrid-ARQ protocol, there are, as discussed at the beginning of the chapter, benefits of having a second-level retransmission mechanism as a complement. By inspecting the sequence numbers of the received PDUs, missing PDUs can be detected and a retransmission requested from the transmitting side.

RLC acknowledged mode in NR is similar to its counterpart in LTE with one exception—reordering to ensure in-sequence delivery is not supported in NR. Removing in-sequence delivery from the RLC also helps reduce the overall latency as later packets do not have to wait for retransmission of an earlier missing packet before being delivered to higher layers, but can be forwarded immediately. This also leads to reduced buffering requirements positively impacting the amount of memory used for RLC buffering. In LTE, which does support in-sequence delivery from the RLC protocol, an RLC SDU cannot be forwarded to higher layers unless all previous SDUs have been correctly received. A single missing SDU, for example, due to a momentary interference burst, can thus block delivery of subsequent SDUs for quite some time, even if those SDUs would be useful to the application, a property which is clearly not desirable in a system targeting very low latency.

In acknowledged mode, the RLC entity is bidirectional—that is, data may flow in both directions between the two peer entities. This is necessary as the reception of PDUs needs to be acknowledged back to the entity that transmitted those PDUs. Information about missing PDUs is provided by the receiving end to the transmitting end in the form of so-called *status reports*. Status reports can either be transmitted autonomously by the receiver or requested by the transmitter. To keep track of the PDUs in transit, the sequence number in the header is used.

Both RLC entities maintain two windows in acknowledged mode, the transmission and reception windows, respectively. Only PDUs in the transmission window are eligible for transmission; PDUs with sequence number below the start of the window have already been acknowledged by the receiving RLC. Similarly, the receiver only accepts PDUs with sequence numbers within the reception window. The receiver also discards any duplicate PDUs as only one copy of each SDU should be delivered to higher layers.

The operation of the RLC with respect to retransmissions is perhaps best understood by the simple example in Fig. 13.10, where two RLC entities are illustrated, one in the transmitting node and one in the receiving node. When operating in acknowledged mode, as assumed below, each RLC entity has both transmitter and receiver functionality, but in this example only one of the directions is discussed as the other direction is identical. In the example, PDUs numbered from n to $n+4$ are awaiting transmission in the transmission buffer. At time t_0, PDUs with sequence number up to and including n have been transmitted and correctly received, but only PDUs up to and including $n-1$ have been acknowledged by the receiver. As seen in the figure, the transmission window starts from n, the first not-yet-acknowledged PDU, while the reception window starts from $n+1$, the next PDU expected to be received. Upon reception of a PDU n, the SDU is reassembled and delivered to higher layers, that is, the PDCP. For a PDU containing a complete SDU, reassembly is simply header removal, but in the case of a segmented SDU, the SDU cannot be delivered until PDUs carrying all the segments have been received.

The transmission of PDUs continues and, at time t_1, PDUs $n+1$ and $n+2$ have been transmitted but, at the receiving end, only PDU $n+2$ has arrived. As soon as a complete SDU is received, it is delivered to higher layers, hence PDU $n+2$ is forwarded to the PDCP layer without waiting for the missing PDU $n+1$. One reason PDU $n+1$ is missing could be that it is under retransmission by the hybrid-ARQ protocol and therefore has not yet been delivered from the hybrid-ARQ to the RLC. The transmission window remains unchanged compared to the previous figure, as none of the PDUs n and higher have been acknowledged by the receiver. Hence, any of these PDUs may need to be retransmitted as the transmitter is not aware of whether they have been received correctly or not.

The reception window is not updated when PDU $n+2$ arrives, the reason being the missing PDU $n+1$. Instead the receiver starts a timer, the *t-Reassembly* timer. If the missing PDU $n+1$ is not received before the timer expires, a retransmission is requested. Fortunately, in this example, the missing PDU arrives from the hybrid-ARQ protocol at time t_2, before the timer expires. The reception window is advanced and the reassembly timer is stopped as the missing PDU has arrived. PDU $n+1$ is delivered for reassembly into SDU $n+1$.

Duplicate detection is also the responsibility of the RLC, using the same sequence number as used for retransmission handling. If PDU $n+2$ arrives again (and is within the reception window), despite it having already been received, it is discarded.

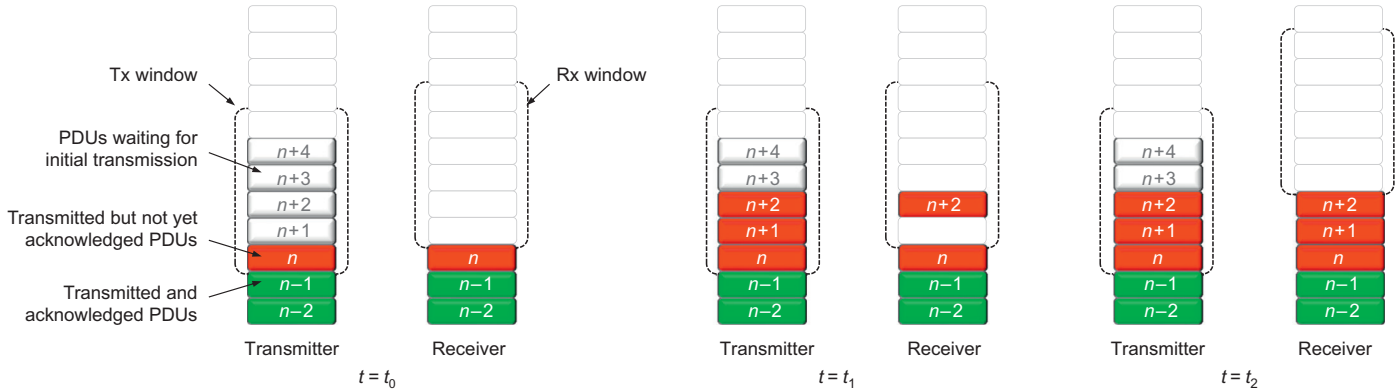

FIGURE 13.10

SDU delivery in acknowledged mode.

The transmission continues with PDUs $n+3$, $n+4$, and $n+5$, as shown in Fig. 13.11. At time t_3, PDUs up to $n+5$ have been transmitted. Only PDU $n+5$ has arrived and PDUs $n+3$ and $n+4$ are missing. Similar to the case above, this causes the reassembly timer to start. However, in this example no PDUs arrive prior to the expiration of the timer. The expiration of the timer at time t_4 triggers the receiver to send a control PDU containing a status report, indicating the missing PDUs, to its peer entity. Control PDUs have higher priority than data PDUs to avoid the status reports being unnecessarily delayed and negatively impacting the retransmission delay. Upon receipt of the status report at time t_5, the transmitter knows that PDUs up to $n+2$ have been received correctly and the transmission window is advanced. The missing PDUs $n+3$ and $n+4$ are retransmitted and, this time, correctly received.

Finally, at time t_6, all PDUs, including the retransmissions, have been delivered by the transmitter and successfully received. As $n+5$ was the last PDU in the transmission buffer, the transmitter requests a status report from the receiver by setting a flag in the header of the last RLC data PDU. Upon reception of the PDU with the flag set, the receiver will respond by transmitting the requested status report, acknowledging all PDUs up to and including $n+5$. Reception of the status report by the transmitter causes all the PDUs to be declared as correctly received and the transmission window is advanced.

Status reports can, as mentioned earlier, be triggered for multiple reasons. However, to control the amount of status reports and to avoid flooding the return link with an excessive number of status reports, it is possible to use a status prohibit timer. With such a timer, status reports cannot be transmitted more often than once per time interval as determined by the timer.

The example above basically assumed each PDU carrying a non-segmented SDU. Segmented SDUs are handled the same way, but an SDU cannot be delivered to the PDCP protocol until all the segments have been received. Status reports and retransmissions operate on individual segments; only the missing segment of a PDU needs to be retransmitted.

In the case of a retransmission, all RLC PDUs may not fit into the transport block size scheduled for the RLC retransmission. Resegmentation following the same principle as the original segmentation is used in this case.

13.3 PDCP

The *Packet Data Convergence Protocol* (PDCP) is responsible for:

- Header compression;
- Ciphering and integrity protection;
- Routing and duplication for split bearers; and
- Retransmission, reordering, and SDU discard.

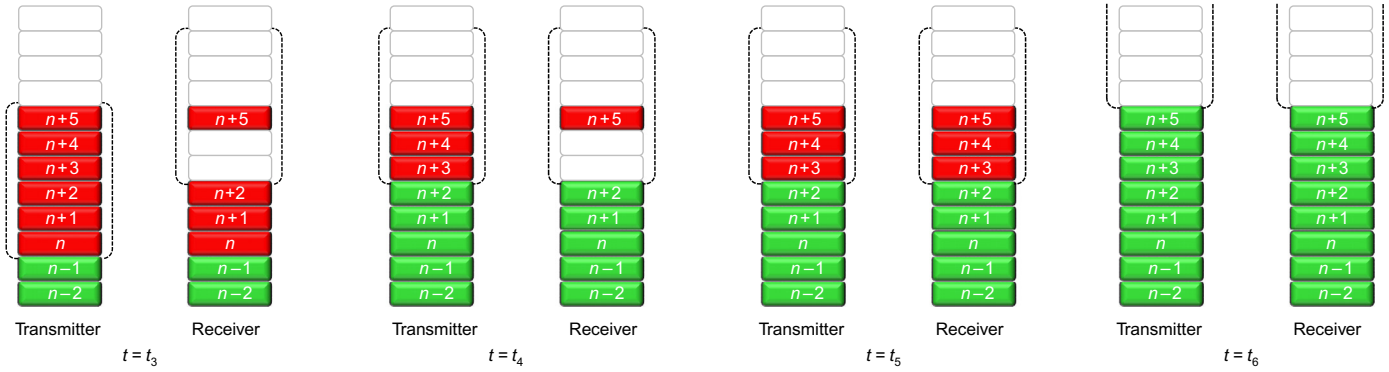

FIGURE 13.11

Retransmission of missing PDUs.

Header compression, with the corresponding decompression functionality at the receiver side, can be configured and serves the purpose of reducing the number of bits transmitted over the radio interface. Especially for small payloads, such as voice-over-IP and TCP acknowledgments, the size of an uncompressed IP header is in the same range as the payload itself, 40 bytes for IP v4 and 60 bytes for IP v6, and can account for around 60% of the total number of bits sent. Compressing this header to a couple of bytes can therefore increase the spectral efficiency by a large amount. The header compression scheme in NR is based on Robust Header Compression (ROHC) [38], a standardized header-compression framework also used for several other mobile-communication technologies, for example, LTE. Multiple compression algorithms, denoted profiles, are defined, each specific to the particular network layer and transport layer protocol combination such as TCP/IP and RTP/UDP/IP. Header compression is developed to compress IP packets. Hence it is applied to the data part only and not the SDAP header (if present).

Integrity protection ensures that the data originate from the correct source and ciphering protects against eavesdropping. PDCP is responsible for both these functions, if configured. Integrity protection and ciphering are used for both the data plane and the control plane and applied to the payload only and not the PDCP control PDUs or SDAP headers.

For dual connectivity and split bearers (see Chapter 6, for a more in-depth discussion on dual connectivity), PDCP can provide routing and duplication functionality. With dual connectivity, some of the radio bearers are handled by the master cell group, while others are handled by the secondary cell group. There is also a possibility to split a bearer across both cell groups. The routing functionality of the PDCP is responsible for routing the data flows for the different bearers to the correct cell groups, as well as handling flow control between the central unit (gNB-CU) and distributed unit (gNB-DU) in the case of a split gNB.

Duplication implies that the same data can be transmitted on two separate logical channels where configuration ensures that the two logical channels are mapped to different carriers. This can be used in combination with carrier aggregation or dual connectivity to provide additional diversity. If multiple carriers are used to transmit the same data, the likelihood that reception of the data on at least one carrier is correct increases. If multiple copies of the same SDU are received, the receiving-side PDCH discards the duplicates. This results in selection diversity which can be essential to providing very high reliability.

Retransmission functionality, including the possibility for reordering to ensure in-sequence delivery, is also part of the PDCP. A relevant question is why the PDCP is capable of retransmissions when there are two other retransmission functions in lower layers, the RLC ARQ and the MAC hybrid-ARQ functions. One reason is inter-gNB handover. Upon handover, undelivered downlink data packets will be forwarded by the PDCP from the old gNB to the new gNB. In this case, a new RLC entity (and hybrid-ARQ entity) is established in the new gNB and the RLC status is lost. The PDCP retransmission functionality ensures that no packets

are lost as a result of this handover. In the uplink, the PDCP entity in the device will handle retransmission of all uplink packets not yet delivered to the gNB as the hybrid-ARQ buffers are flushed upon handover.

In-sequence delivery is not ensured by the RLC to reduce the overall latency. In many cases, rapid delivery of the packets is more important than guaranteed in-sequence delivery. However, if in-sequence delivery is important, the PDCP can be configured to provide this.

Retransmission and in-sequence delivery, if configured, is jointly handled in the same protocol, which operates similarly to the RLC ARQ protocol except that no segmentation is supported. A so-called count value is associated with each SDU, where the count is a combination of the PDCP sequence number and the hyper-frame number. The count value is used to identify lost SDUs and request retransmission, as well as reorder received SDUs before delivery to upper layers is reordering is configured. Reordering basically buffers a received SDU and does not forward it to higher layers until all lower-numbered SDUs have been delivered. Referring to Fig. 13.10, this would be similar to not delivering SDU $n + 2$ until $n + 1$ has been successfully received and delivered. There is also a possibility to configure a discard timer for each PDCP SDU; when the timer expires the corresponding SDU is discarded and not transmitted.