CHAPTER

8 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 is discussed in the next chapter. However, receiver noise and unpredictable interference variations cannot be counteracted. Therefore, virtually all wireless communications systems employ some form of *forward error correction* (FEC), adding redundancy to the transmitted signal allowing the receiver to correct errors, based on the concept whose roots can be traced to the pioneering work of Claude Shannon in 1948 [7]. In LTE, Turbo coding is used as discussed in Chapter 6.

Despite the error-correcting code, there will be data units received in error. *Hybrid automatic repeat request* (ARQ), first proposed in [18] 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 LTE, there are *two* mechanisms responsible for retransmission handling, namely the MAC and RLC sublayers. Retransmissions of missing or erroneous data units are handled primarily by the hybrid-ARQ mechanism in the MAC layer, complemented by the retransmission functionality of the RLC protocol. The reasons for having a two-level retransmission structure can be found in the trade-off between fast and reliable feedback of the status reports. The hybrid-ARQ mechanism targets very fast retransmissions and, consequently, feedback on success or failure of the decoding attempt is provided to the transmitter after each received transport block. 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 power. Keeping the cost reasonable typically results in a feedback error rate of around 1%, which results in a hybrid-ARQ residual error rate of a similar order. Such an error rate is in many cases far too high; high data rates with TCP may require virtually error-free delivery of packets to the TCP protocol layer. As an example, for sustainable data rates exceeding 100 Mbit/s, a packet-loss probability less than 10^{-5} is required [35]. 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 better is relatively

4G, LTE-Advanced Pro and The Road to 5G. http://dx.doi.org/10.1016/B978-0-12-804575-6.00008-X

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. As the MAC and RLC protocol layers are located in the same network node, a tight interaction between the two protocols is possible. Hence, to some extent, the combination of the two can be viewed as *one* retransmission mechanism with *two* feedback channels. However, note that, as discussed in Chapter 4 and illustrated in Figure 8.1, the RLC operates per logical channel, while the hybrid-ARQ operates per transport channel (that is, per component carrier). One hybrid-ARQ entity may therefore retransmit data belonging to multiple logical channels.

In the following section, the principles behind the hybrid-ARQ and RLC protocols are discussed in more detail.

8.1 HYBRID ARQ WITH SOFT COMBINING

The hybrid-ARQ operation described in the preceding section discards erroneously received packets and requests retransmission. However, despite it was not 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

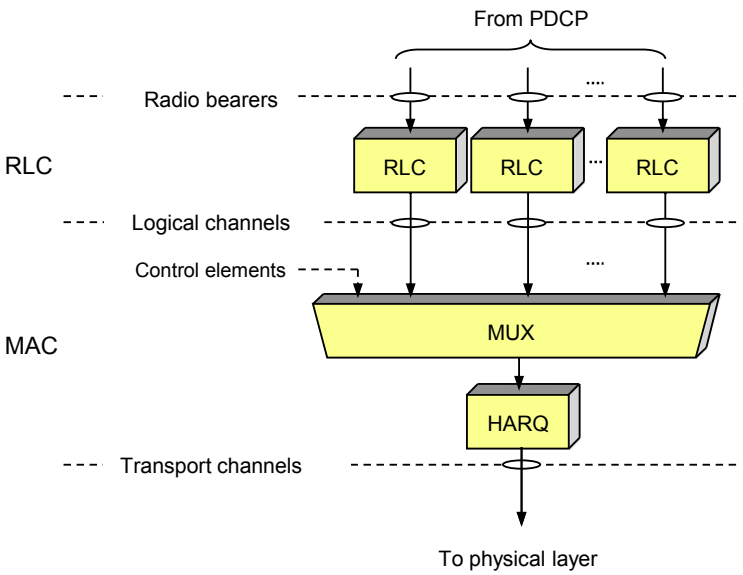

FIGURE 8.1

RLC and hybrid-ARQ retransmission mechanisms in LTE.

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.

The hybrid-ARQ functionality spans both the physical layer and the MAC layer; generation of different redundancy versions at the transmitter as well as the soft combining at the receiver are handled by the physical layer, while the hybrid-ARQ protocol is part of the MAC layer.

The basis for the LTE hybrid-ARQ mechanism is 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. LTE therefore applies *multiple* stop-and-wait processes operating in parallel such that, while waiting for acknowledgment from one process, the transmitter can transmit data to another hybrid-ARQ process. This is illustrated in Figure 8.3; while processing the data received in the first hybrid-ARQ process the receiver can continue to receive using the second process and so on. 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.

There is one hybrid-ARQ entity per device. Spatial multiplexing, where two transport blocks can be transmitted in parallel on the same transport channel as described in Chapter 6, is supported by one hybrid-ARQ entity having two sets of hybrid-ARQ processes with independent hybrid-ARQ acknowledgments. The details for the physical-layer transmission of the downlink and uplink hybrid-ARQ acknowledgments are described in Chapters 6 and 7.

Upon receiving a transport block for a certain hybrid-ARQ process, the receiver makes an attempt to decode the transport block and informs the transmitter about the outcome through a hybrid-ARQ acknowledgment, indicating whether the transport block was correctly decoded or not. The time from reception of data until transmission of the hybrid-ARQ acknowledgment is fixed, hence the transmitter knows from the timing relation which hybrid-ARQ process a received acknowledgment relates to. This is beneficial from an overhead perspective as there is no need to signal the process number along with the acknowledgment.

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, retransmission in any hybrid-ARQ scheme 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 [19], or *incremental redundancy* (IR), which is used in LTE. With IR, 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 [20,21]. The rate matching functionality of LTE, described in Chapter 6, is used to generate different sets of coded bits as a function of the redundancy version as illustrated in Figure 8.2. In addition to a gain in accumulated received E_b/N_0, IR also results in a coding gain for each retransmission. The gain with IR compared to pure energy accumulation (chase combining) is larger for high initial code rates [22]. Furthermore, as shown in [23], the performance gain of IR compared to chase combining can also depend on the relative power difference between the transmission attempts.

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. One example here is Turbo codes, where the systematic bits are of higher importance than the parity bits. Hence, the initial transmission should at least include the systematic bits and some parity bits. In the retransmission(s), parity bits not in the initial transmission can be included.

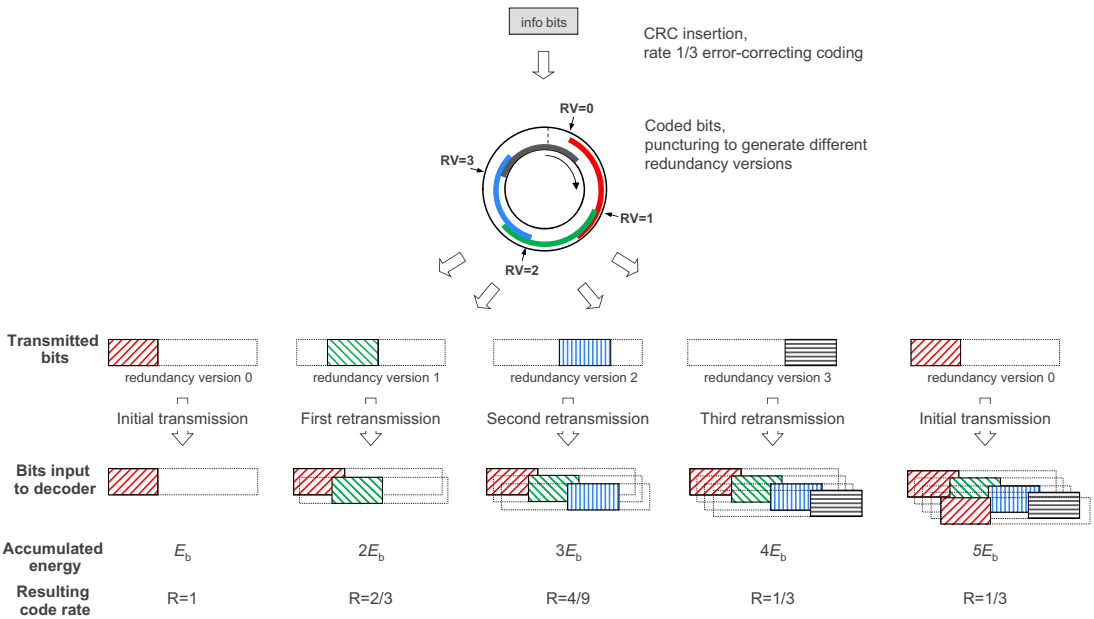

FIGURE 8.2

Example of incremental redundancy.

This is the background to why systematic bits are inserted first in the circular buffer discussed in Chapter 6.

Hybrid ARQ with soft combining, regardless of whether chase combining or IR 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. Therefore, an explicit *new-data indicator* is included for each of the one or two scheduled transport blocks as part of the scheduling information transmitted in the downlink. The new-data indicator is present in both downlink assignments and uplink grants, although the meaning is slightly different for the two.

For downlink data transmission, 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.

For uplink data transmission, there is also a new-data indicator transmitted on the downlink PDCCH. In this case, 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 eNodeB should perform soft combining).

The use of multiple hybrid-ARQ processes operating in parallel can result in data being delivered from the hybrid-ARQ mechanism out of sequence. For example, transport block 5 in Figure 8.3 was successfully decoded before transport block 1, which required two retransmissions. Out-of-sequence delivery can also occur in the case of carrier aggregation, where transmission of a transport block on one component carrier could be successful while a retransmission is required on another component carrier. To handle out-of-sequence delivery

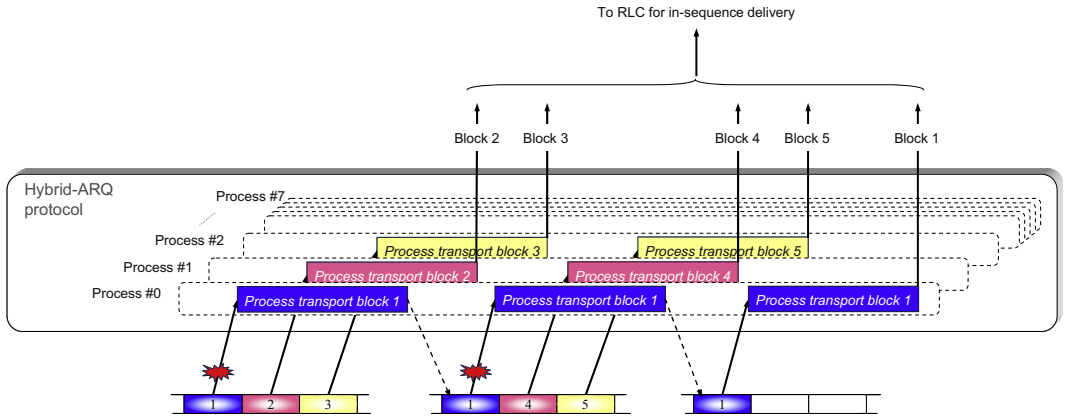

FIGURE 8.3

Multiple parallel hybrid-ARQ processes forming one hybrid-ARQ entity.

from the hybrid-ARQ protocol, the RLC protocol includes an in-sequence-delivery mechanism, as described in Section 8.2.

Hybrid-ARQ protocols can be characterized as synchronous versus asynchronous, related to the flexibility in the time domain, as well as adaptive versus nonadaptive, related to the flexibility in the frequency domain:

- An *asynchronous* hybrid-ARQ protocol implies that retransmissions can occur at any time, whereas a *synchronous* protocol implies that retransmissions occur at a fixed time after the previous transmission (see Figure 8.4). The benefit of a synchronous protocol is

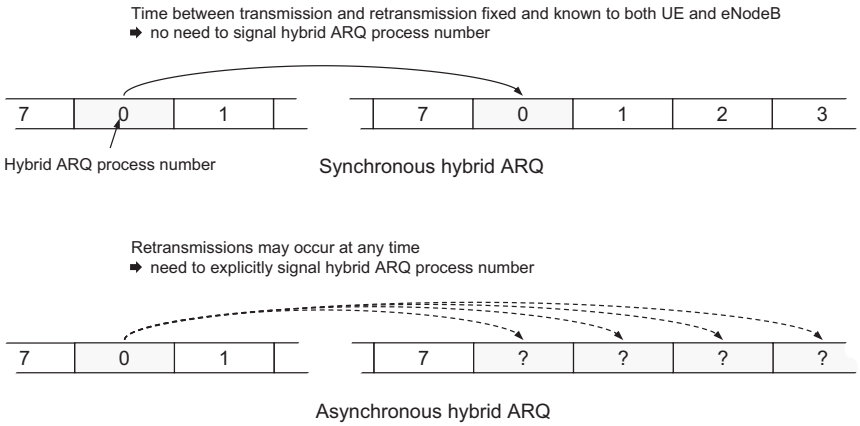

FIGURE 8.4

Synchronous and asynchronous hybrid ARQ.

that there is no need to explicitly signal the hybrid-ARQ process number as this information can be derived from the subframe number. On the other hand, an asynchronous protocol allows for more flexibility in the scheduling of retransmissions.

- An *adaptive* hybrid-ARQ protocol implies that the frequency location and possibly also the more detailed transmission format can be changed between retransmissions. A *nonadaptive protocol*, in contrast, implies that the retransmission must occur at the same frequency resources and with the same transmission format as the initial transmission.

In the case of LTE, asynchronous adaptive hybrid ARQ is used for the downlink. For the uplink, synchronous hybrid ARQ is used. Typically, the retransmissions are nonadaptive, but there is also the possibility to use adaptive retransmissions as a complement.

8.1.1 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. Hence, the downlink hybrid-ARQ protocol is *asynchronous* and *adaptive*. The support for asynchronous and adaptive hybrid ARQ for the LTE downlink is motivated by the need to avoid collisions with, for example, transmission of system information and MBSFN subframes. Instead of dropping a retransmission that otherwise would collide with MBSFN subframes or transmission of system information, the eNodeB can move the retransmission in time and/or frequency to avoid the overlap in resources.

Support for soft combining is, as described in the introduction, provided through an explicit new-data indicator, toggled for each new transport block. In addition to the new-data indicator, hybrid-ARQ-related downlink control signaling consists of the hybrid-ARQ process number (three bits for FDD, four bits for TDD) and the redundancy version (two bits), both explicitly signaled in the scheduling assignment for each downlink transmission.

Downlink spatial multiplexing implies, as already mentioned, transmission of two transport blocks in parallel on a component carrier. To provide the possibility to retransmit only one of the transport blocks, which is beneficial as error events for the two transport blocks can be fairly uncorrelated, each transport block has its own separate new-data indicator and redundancy-version indication. However, there is no need to signal the process number separately as each process consists of two sub-processes in the case of spatial multiplexing or, expressed differently, once the process number for the first transport block is known, the process number for the second transport block is given implicitly.

Transmissions on downlink component carriers are acknowledged independently. On each component carrier, transport blocks are acknowledged by transmitting one or two bits on the uplink, as described in Chapter 7. In the absence of spatial multiplexing, there is only a single transport block within a TTI and consequently only a single acknowledgment bit is required in response. However, if the downlink transmission used spatial multiplexing, there are two transport blocks per TTI, each requiring its own hybrid-ARQ acknowledgment bit. The total

number of bits required for hybrid-ARQ acknowledgments thus depends on the number of component carriers and the transmission mode for each of the component carriers. As each downlink component carrier is scheduled separately from its own PDCCH, the hybrid-ARQ process numbers are signaled independently for each component carrier.

The device should not transmit a hybrid-ARQ acknowledgment in response to reception of system information, paging messages, and other broadcast traffic. Hence, hybrid-ARQ acknowledgments are only sent in the uplink for "normal" unicast transmission.

8.1.2 UPLINK HYBRID ARQ

Shifting the focus to the uplink, a difference compared to the downlink case is the use of synchronous, nonadaptive operation as the basic principle of the hybrid-ARQ protocol, motivated by the lower overhead compared to an asynchronous, adaptive structure. Hence, uplink retransmissions always occur at an a priori known subframe; in the case of FDD operation uplink retransmissions occur eight subframes after the prior transmission attempt for the same hybrid-ARQ process. The set of resource blocks used for the retransmission on a component carrier is identical to the initial transmission. Thus, the only control signaling required in the downlink for a retransmission is a retransmission command, transmitted on the PHICH as described in Chapter 6. In the case of a negative acknowledgment on the PHICH, the data are retransmitted.

Spatial multiplexing is handled in the same way as in the downlink—two transport blocks transmitted in parallel on the uplink, each with its own modulation-and-coding scheme and new-data indicator but sharing the same hybrid-ARQ process number. The two transport blocks are individually acknowledged and hence two bits of information are needed in the downlink to acknowledge an uplink transmission using spatial multiplexing. TDD is another example, as is discussed later, where uplink transmissions in different subframes may need to be acknowledged in the same downlink subframe.[1] Thus, multiple PHICHs may be needed as each PHICH is capable of transmitting a single bit only.

Despite the fact that the basic mode of operation for the uplink is synchronous, nonadaptive hybrid ARQ, there is also the possibility to operate the uplink hybrid ARQ in a synchronous, *adaptive* manner, where the resource-block set and modulation-and-coding scheme for the retransmissions are changed. Although nonadaptive retransmissions are typically used due to the very low overhead in terms of downlink control signaling, adaptive retransmissions are sometimes useful to avoid fragmenting the uplink frequency resource or to avoid collisions with random-access resources. This is illustrated in Figure 8.5. A device is scheduled for an initial transmission in subframe n; a transmission that is not correctly received and consequently a retransmission is required in subframe $n + 8$ (assuming FDD; for TDD the timing depends on the uplink−downlink allocation, as discussed later). With nonadaptive hybrid ARQ, the retransmissions occupy the same part of the uplink spectrum as

[1] A third example is carrier aggregation, see Chapter 12.

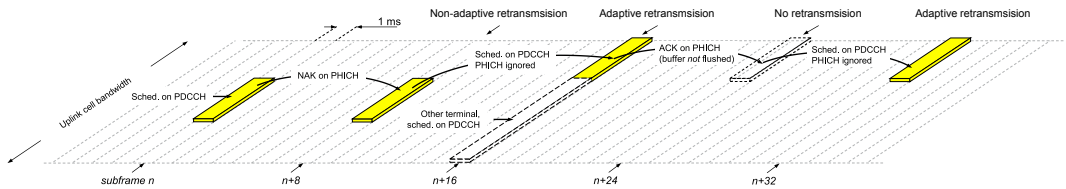

FIGURE 8.5

Nonadaptive and adaptive hybrid-ARQ operation.

the initial transmission. Hence, in this example the spectrum is fragmented, which limits the bandwidth available to another device (unless the other device is capable of multi-cluster transmission). In subframe $n + 16$, an example of an adaptive retransmission is found; to make room for another device to be granted a large part of the uplink spectrum, the retransmission is moved in the frequency domain. It should be noted that the uplink hybrid-ARQ protocol is still synchronous—that is, a retransmission should always occur eight subframes after the previous transmission.

The support for both adaptive and nonadaptive hybrid ARQ is realized by *not* flushing the transmission buffer when receiving a positive hybrid-ARQ acknowledgment on PHICH for a given hybrid-ARQ process. Instead, the actual control of whether data should be retransmitted or not is done by the new-data indicator included in the uplink scheduling grant sent on the PDCCH. The new-data indicator is toggled for each new transport block. If the new-data indicator is toggled, the device flushes the transmission buffer and transmits a new-data packet. However, if the new-data indicator does not request transmission of a new transport block, the previous transport block is retransmitted. Hence, clearing of the transmission buffer is not handled by the PHICH but by the PDCCH as part of the uplink grant. The negative hybrid-ARQ acknowledgment on the PHICH should preferably be seen as a single-bit scheduling grant for retransmissions where the set of bits to transmit and all the resource information are known from the previous transmission attempt. In Figure 8.5 an example of postponing a transmission is seen in subframe $n + 24$. The device has received a positive acknowledgment and therefore does *not* retransmit the data. However, the transmission buffer is not flushed, which later is exploited by an uplink grant requesting retransmission in subframe $n + 32$.

A consequence of the above-mentioned method of supporting both adaptive and nonadaptive hybrid ARQ is that the PHICH and PDCCH related to the same uplink subframe have the same timing. If this were not the case, the complexity would increase as the device would not know whether to obey the PHICH or wait for a PDCCH overriding the PHICH.

As explained earlier, the new-data indicator is explicitly transmitted in the uplink grant. However, unlike the downlink case, the redundancy version is *not* explicitly signaled for each retransmission. With a single-bit acknowledgment on the PHICH, this is not possible. Instead, as the uplink hybrid-ARQ protocol is synchronous, the redundancy version follows a

predefined pattern, starting with zero when the initial transmission is scheduled by the PDCCH. Whenever a retransmission is requested by a negative acknowledgment on the PHICH, the next redundancy version in the sequence is used. However, if a retransmission is explicitly scheduled by the PDCCH overriding the PHICH, then there is the potential to affect the redundancy version to use. Grants for retransmissions use the same format as ordinary grants (for initial transmissions). One of the information fields in an uplink grant is, as described in Chapter 6, the modulation-and-coding scheme. Of the 32 different combinations this five-bit field can take, three of them are reserved. Those three combinations represent different redundancy versions; hence, if one of these combinations is signaled as part of an uplink grant indicating a retransmission, the corresponding redundancy version is used for the transmission. The transport-block size is already known from the initial transmission as it, by definition, cannot change between retransmission attempts. In Figure 8.5, the initial transmission in subframe n uses the first redundancy version in sequence as the transport-block size must be indicated for the initial transmission. The retransmission in subframe $n + 8$ uses the next redundancy version in the sequence, while the explicitly scheduled retransmission in subframe $n + 16$ can use any redundancy scheme as indicated on the PDCCH.

Whether to exploit the possibility to signal an arbitrary redundancy version in a retransmission scheduled by the PDCCH is a trade-off between incremental-redundancy gain and robustness. From an incremental-redundancy perspective, changing the redundancy value between retransmissions is typically beneficial to fully exploit the gain from IR. However, as the modulation-and-coding scheme is normally not indicated for uplink retransmissions, as either the single-bit PHICH is used or the modulation-and-coding field is used to explicitly indicate a new redundancy version, it is implicitly assumed that the device did not miss the initial scheduling grant. If this is the case, it is necessary to explicitly indicate the modulation-and-coding scheme, which also implies that the first redundancy version in the sequence is used.

8.1.3 HYBRID-ARQ TIMING

The receiver must know to which hybrid-ARQ process a received acknowledgment is associated. This is handled by the timing of the hybrid-ARQ acknowledgment being used to associate the acknowledgment with a certain hybrid-ARQ process; the timing relation between the reception of data in the downlink and transmission of the hybrid-ARQ acknowledgment in the uplink (and vice versa) is fixed. From a latency perspective, the time between the reception of downlink data at the device and transmission of the hybrid-ARQ acknowledgment in the uplink should be as short as possible. At the same time, an unnecessarily short time would increase the demand on the device processing capacity and a trade-off between latency and implementation complexity is required. The situation is similar for uplink data transmissions. For LTE, this trade-off led to the decision to have eight hybrid-ARQ processes per component carrier in both uplink and downlink for FDD. For TDD, the number of processes depends on the uplink−downlink allocation, as discussed later.

8.1.3.1 Hybrid-ARQ Timing for FDD

Starting with the FDD case, transmission of acknowledgments in the uplink in response to downlink data transmission is illustrated in Figure 8.6. Downlink data on the DL-SCH is transmitted to the device in subframe n and received by the device, after the propagation delay T_p, in subframe n. The device attempts to decode the received signal, possibly after soft combining with a previous transmission attempt, and transmits the hybrid-ARQ acknowledgment in uplink subframe $n + 4$ (note that the start of an uplink subframe at the device is offset by T_{TA} relative to the start of the corresponding downlink subframe at the device as a result of the timing-advance procedure described in Section 7.6). Upon reception of the hybrid-ARQ acknowledgment, the eNodeB can, if needed, retransmit the downlink data in subframe $n + 8$. Thus, eight hybrid-ARQ processes are used—that is, the hybrid-ARQ round-trip time is 8 ms.

The description in the previous paragraph, as well as the illustration in Figure 8.6, describe the timing for downlink data transmission—that is, data on DL-SCH and acknowledgments on PUCCH (or PUSCH). However, the timing of uplink data transmission—that is, data on PUSCH and acknowledgments on PHICH—is identical, namely uplink data transmission in subframe n results in a PHICH transmission in subframe $n + 4$. This is also the same timing relation as for uplink scheduling grants in general (as described in Section 9.3) and allows a scheduling grant on the PDCCH to override the PHICH, as shown in Figure 8.5.

In Figure 8.6 it is seen that the processing time available to the device, T_{UE}, depends on the value of the timing advance or, equivalently, on the device-to-base-station distance. As a device must operate at any distance up to the maximum size supported by the specifications, the device must be designed such that it can handle the worst-case scenario. LTE is designed to handle at least 100 km, corresponding to a maximum timing advance of 0.67 ms. Hence, there is approximately 2.3 ms left for the device processing, which is considered a reasonable

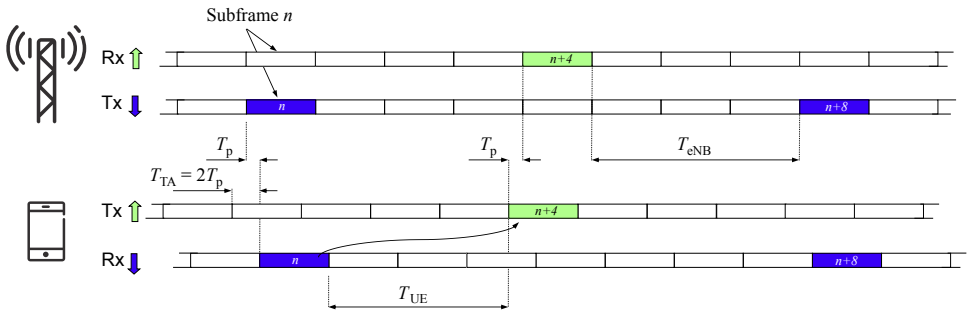

FIGURE 8.6

Timing relation between downlink data in subframe n and uplink hybrid-ARQ acknowledgment in subframe $n + 4$ for FDD.

trade-off between the processing requirements imposed on a device and the associated delays.

For the eNodeB, the processing time available, denoted as T_{eNB}, is 3 ms and thus of the same order as for the device. In the case of downlink data transmission, the eNodeB performs scheduling of any retransmissions during this time and, for uplink data transmission, the time is used for decoding of the received signal. The timing budget is thus similar for the device and the eNodeB, which is motivated by the fact that, although the eNodeB typically has more processing power than a device, it also has to serve multiple devices and to perform the scheduling operation.

Having the same number of hybrid-ARQ processes in uplink and downlink is beneficial for half-duplex FDD operation, discussed in Chapter 9. By proper scheduling, the uplink hybrid-ARQ transmission from the device will coincide with transmission of uplink data and the acknowledgments related to the reception of uplink data will be transmitted in the same subframe as the downlink data. Thus, using the same number of hybrid-ARQ processes in uplink and downlink results in a 50:50 split between transmission and reception for a half-duplex device.

8.1.3.2 Hybrid-ARQ Timing for TDD

For TDD operation, the time relation between the reception of data in a certain hybrid-ARQ process and the transmission of the hybrid-ARQ acknowledgment depends on the uplink–downlink allocation. An uplink hybrid-ARQ acknowledgment can only be transmitted in an uplink subframe and a downlink acknowledgment only in a downlink subframe. Furthermore, starting from release 11, different uplink–downlink allocations can be used on component carriers operating in different frequency bands. Therefore, the hybrid-ARQ timing relations for TDD are more intricate than their FDD counterparts.

For simplicity, consider the situation when the same uplink–downlink allocation is used across all component carriers (the case of different allocations is described later). Since the minimum processing time required in the device and the eNodeB remains the same in FDD and TDD as the same turbo decoders are used and the scheduling decisions are similar, acknowledgments cannot be transmitted earlier than in FDD. Therefore, for TDD the acknowledgment of a transport block in subframe n is transmitted in subframe $n + k$, where $k \geq 4$ and is selected such that $n + k$ is an uplink subframe when the acknowledgment is to be transmitted from the device (on PUCCH or PUSCH) and a downlink subframe when the acknowledgment is transmitted from the eNodeB (on PHICH). The value of k depends on the uplink–downlink configuration, as shown in Table 8.1 for both downlink and uplink transmissions. From the table it is seen that, as a consequence of the uplink–downlink configuration, the value of k is sometimes larger than the FDD value, $k = 4$. For example, assuming configuration 2, an uplink transmission on PUSCH received in subframe 2 should be acknowledged on PHICH in subframe $2 + 6 = 8$. Similarly, for the same configuration, downlink transmission on PDSCH in subframe 0 should be acknowledged on PUCCH (or PUSCH) in subframe $0 + 7 = 7$.

Table 8.1 Number of Hybrid-ARQ Processes and Acknowledgment Timing *k* for Different TDD Configurations

Configuration (DL:UL)	Downlink											Uplink										
	Proc	PDSCH Reception in Subframe *n*										Proc	PUSCH Reception in Subframe *n*									
		0	1	2	3	4	5	6	7	8	9		0	1	2	3	4	5	6	7	8	9
0 (2:3)	4	4	6	–	–	–	4	6	–	–	–	6	–	–	4	7	6	–	–	4	7	6
1 (3:2)	7	7	6	–	–	4	7	6	–	–	4	4	–	–	4	6	–	–	–	4	6	–
2 (4:1)	10	**7**	6	–	**4**	8	7	6	–	4	8	2	–	–	6	–	–	–	–	6	–	–
3 (7:3)	9	4	11	–	–	–	7	6	6	5	5	3	–	–	6	6	6	–	–	–	–	–
4 (8:2)	12	12	11	–	–	8	7	7	6	5	4	2	–	–	6	6	–	–	–	–	–	–
5 (9:1)	15	12	11	–	9	8	7	6	5	4	13	1	–	–	6	–	–	–	–	–	–	–
6 (5:5)	6	7	7	–	–	–	7	7	–	–	5	6	–	–	4	6	6	–	–	4	7	–

From the table it is also seen that the number of hybrid-ARQ processes used for TDD depends on the uplink–downlink configuration, implying that the hybrid-ARQ round-trip time is configuration dependent for TDD (actually, it may even vary between subframes, as seen in Figure 8.7). For the downlink-heavy configurations 2, 3, 4, and 5, the number of downlink hybrid-ARQ processes is larger than for FDD. The reason is the limited number of uplink subframes available, resulting in k-values well beyond 4 for some subframes (Table 8.1).

The PHICH timing in TDD is identical to the timing when receiving an uplink grant, as described in Section 9.3. The reason is the same as in FDD, namely to allow a PDCCH uplink grant to override the PHICH in order to implement adaptive retransmissions, as illustrated in Figure 8.5.

The uplink–downlink allocation for TDD has implications on the number of transport blocks to acknowledge in a single subframe. For FDD there is always a one-to-one relation between uplink and downlink subframes. Hence, in absence of carrier aggregation, a subframe only needs to carry acknowledgments for one subframe in the other direction. For TDD, in contrast, there is not necessarily a one-to-one relation between uplink and downlink subframes. This can be seen in the possible uplink–downlink allocations described in Chapter 5.

For uplink transmissions using UL-SCH, each uplink transport block is acknowledged individually by using the PHICH. Hence, in uplink-heavy asymmetries (uplink–downlink configuration 0), the device may need to receive two hybrid-ARQ acknowledgments in downlink subframes 0 and 5 in the absence of uplink spatial multiplexing; for spatial multiplexing up to four acknowledgments per component carrier are needed. There are also some subframes where no PHICH will be transmitted. The amount of PHICH groups may therefore vary between subframes for TDD.

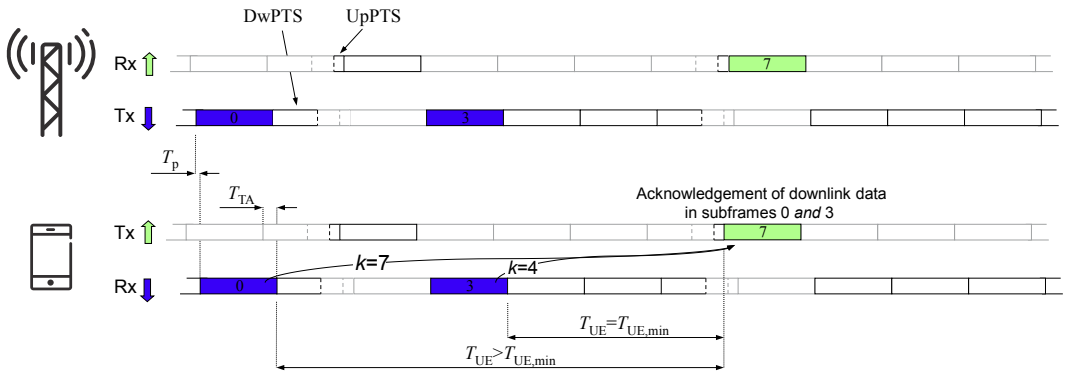

FIGURE 8.7

Example of timing relation between downlink data and uplink hybrid-ARQ acknowledgment for TDD (configuration 2).

For downlink transmissions, there are some configurations where DL-SCH receipt in multiple downlink subframes needs to be acknowledged in a single uplink subframe, as illustrated in Figure 8.7 (the illustration corresponds to the two entries shown in bold in Table 8.1). Two different mechanisms to handle this are provided in TDD: multiplexing and bundling.

Multiplexing implies that independent acknowledgments for each of the received transport blocks are fed back to the eNodeB. This allows independent retransmission of erroneous transport blocks. However, it also implies that multiple bits need to be transmitted from the device, which may limit the uplink coverage. This is the motivation for the bundling mechanism.

Bundling of acknowledgments implies that the outcome of the decoding of downlink transport blocks from multiple downlink subframes can be combined into a single hybrid-ARQ acknowledgment transmitted in the uplink. Only if both of the downlink transmissions in subframes 0 and 3 in the example in Figure 8.7 are correctly decoded will a positive acknowledgment be transmitted in uplink subframe 7.

Combining acknowledgments related to multiple downlink transmissions into a single uplink message assumes that the device has not missed any of the scheduling assignments upon which the acknowledgment is based. Assume, as an example, that the eNodeB scheduled the device in two (subsequent) subframes, but the device missed the PDCCH transmission in the first of the two subframes and successfully decoded the data transmitted in the second subframe. Without any additional mechanism, the device will transmit an acknowledgment based on the assumption that it was scheduled in the second subframe only, while the eNodeB will interpret acknowledgment as the device successfully received both transmissions. To avoid such errors, the *downlink assignment index* (see Section 6.4.6) in the scheduling assignment is used. The downlink assignment index in essence informs the device about the number of transmissions it should base the combined acknowledgment upon. If there is a mismatch between the assignment index and the number of transmissions the device received, the device concludes at least one assignment was missed and transmits no hybrid-ARQ acknowledgment, thereby avoiding acknowledging transmissions not received.

8.2 RADIO-LINK CONTROL

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 Figure 8.8. Multiplexing of several logical channels into a single transport channel is mainly used for priority handling, as described in Section 9.2 in conjunction with downlink and uplink scheduling.

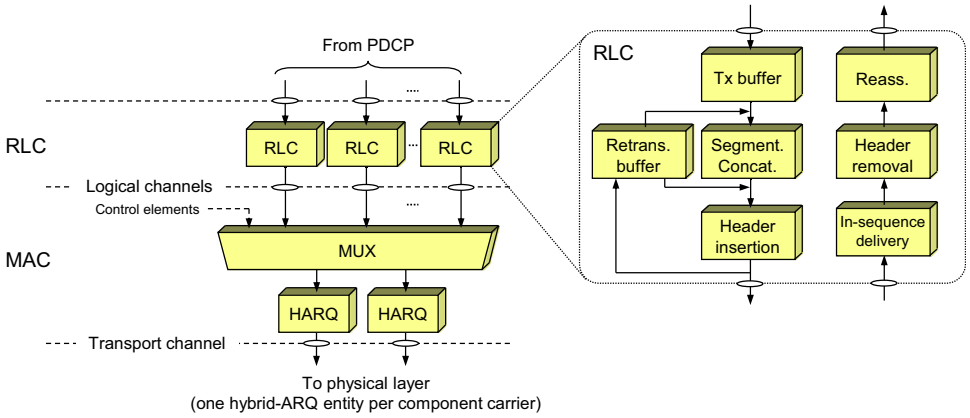

FIGURE 8.8

MAC and RLC structure (single-device view).

There is one RLC entity per logical channel configured for a device, where each RLC entity is responsible for:

- segmentation, concatenation, and reassembly of RLC SDUs;
- RLC retransmission;
- in-sequence delivery and duplicate detection for the corresponding logical channel.

Other noteworthy features of the RLC are: (1) the handling of varying PDU sizes; and (2) the possibility for close interaction between the hybrid-ARQ and RLC protocols. Finally, the fact that there is one RLC entity per logical channel and one hybrid-ARQ entity per component carrier implies that one RLC entity may interact with multiple hybrid-ARQ entities in case of carrier aggregation.

8.2.1 SEGMENTATION, CONCATENATION, AND REASSEMBLY OF RLC SDUS

The purpose of the segmentation and concatenation mechanism is to generate RLC PDUs of appropriate size from the incoming RLC SDUs. One possibility would be to define a fixed PDU size, a size that would result in a compromise. If the size were too large, it would not be possible to support the lowest data rates. Also, excessive padding would be required in some scenarios. A single small PDU size, however, would result in a high overhead from the header included with each PDU. To avoid these drawbacks, which is especially important given the very large dynamic range of data rates supported by LTE, the RLC PDU size varies dynamically.

Segmentation and concatenation of RLC SDUs into RLC PDUs are illustrated in Figure 8.9. The header includes, among other fields, a sequence number, which is used by the

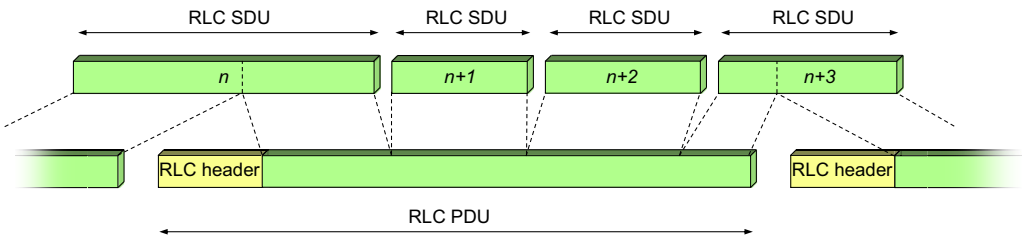

FIGURE 8.9

Generation of RLC PDUs from RLC SDUs.

reordering and retransmission mechanisms. The reassembly function at the receiver side performs the reverse operation to reassemble the SDUs from the received PDUs.

8.2.2 RLC RETRANSMISSION

Retransmission of missing PDUs is one of the main functionalities of the RLC. 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.

Different services have different requirements; for some services (e.g., transfer of a large file), error-free delivery of data is important, whereas for other applications (e.g., 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 segmentation/reassembly, and no in-sequence delivery 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 are 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 feed back status reports as no uplink has been established.
- *Unacknowledged mode* (UM) supports segmentation/reassembly and in-sequence delivery, but not retransmissions. This mode is used when error-free delivery is not required, for example voice-over IP, or when retransmissions cannot be requested, for example broadcast transmissions on MTCH and MCCH using MBSFN.
- *Acknowledged mode* (AM) is the main mode of operation for TCP/IP packet data transmission on the DL-SCH. Segmentation/reassembly, in-sequence delivery, and retransmissions of erroneous data are all supported.

In what follows, the acknowledged mode is described. The unacknowledged mode is similar, with the exception that no retransmissions are done and each RLC entity is unidirectional.

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 transmitter attaches an RLC header to each PDU, including, among other fields, a sequence number.

Both RLC entities maintain two windows, 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 each PDU should be assembled into an SDU only once.

8.2.3 IN-SEQUENCE DELIVERY

In-sequence delivery implies that data blocks are delivered by the receiver in the same order as they were transmitted. This is an essential part of RLC; the hybrid-ARQ processes operate independently and transport blocks may therefore be delivered out of sequence, as seen in Figure 8.3. In-sequence delivery implies that SDU n should be delivered prior to SDU $n + 1$. This is an important aspect as several applications require the data to be received in the same order as it was transmitted. TCP can, to some extent, handle IP packets arriving out of sequence, although with some performance impact, while for some streaming applications in-sequence delivery is essential. The basic idea behind in-sequence delivery is to store the received PDUs in a buffer until all PDUs with lower sequence number have been delivered. Only when all PDUs with lower sequence number have been used for assembling SDUs is the next PDU used. RLC retransmission, provided when operating in acknowledged mode only, operates on the same buffer as the in-sequence delivery mechanism.

8.2.4 RLC OPERATION

The operation of the RLC with respect to retransmissions and in-sequence delivery is perhaps best understood by the simple example in Figure 8.10, where two RLC entities are illustrated, one in the transmitting node and the other 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

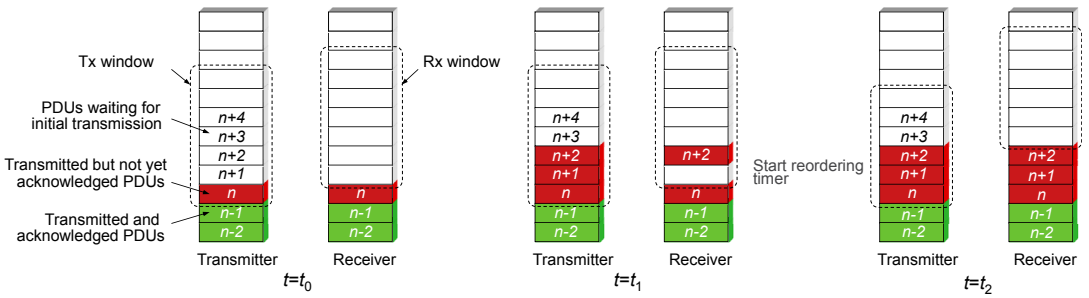

FIGURE 8.10

In-sequence delivery.

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 PDU n, the PDU is forwarded to the SDU reassembly functionality for further processing.

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. One reason for this could be that the missing PDU, $n+1$, is being retransmitted 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. The reception window is not updated when PDU $n+2$ arrives. The reason is that PDU $n+2$ cannot be forwarded to SDU assembly as PDU $n+1$ is missing. Instead, the receiver waits for the missing PDU $n+1$. Waiting for the missing PDU for an infinite time would stall the queue. Hence, the receiver starts a timer, the *reordering timer*, for the missing PDU. If the PDU 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 reordering timer is stopped as the missing PDU has arrived. PDUs $n+1$ and $n+2$ are delivered for reassembly into SDUs.

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

This example illustrates the basic principle behind in-sequence delivery, which is supported by both acknowledged and unacknowledged modes. However, the acknowledged mode of operation also provides retransmission functionality. To illustrate the principles behind this, consider Figure 8.11, which is a continuation of the example mentioned earlier. At time t_3, PDUs up to $n+5$ have been transmitted. Only PDU $n+5$ has arrived and PDUs

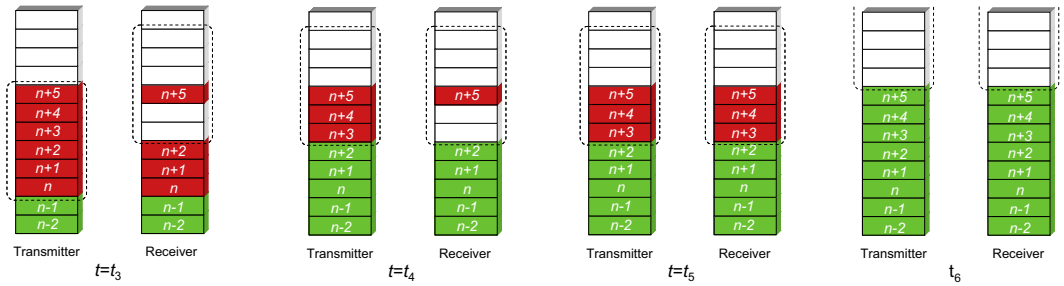

FIGURE 8.11

Retransmission of missing PDUs.

$n + 3$ and $n + 4$ are missing. Similar to the preceding case, this causes the reordering 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 reception 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.

The retransmission was triggered by the reception of a status report in this example. However, as the hybrid-ARQ and RLC protocols are located in the same node, tight interaction between the two is possible. The hybrid-ARQ protocol at the transmitting end could therefore inform the RLC at the transmitting end in case the transport block(s) containing PDUs $n + 3$ and $n + 4$ have failed. The RLC can use this to trigger retransmission of missing PDUs without waiting for an explicit RLC status report, thereby reducing the delays associated with RLC retransmissions.

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.

For the initial transmission, it is relatively straightforward to rely on a dynamic PDU size as a means to handle the varying data rates. However, the channel conditions and the amount of resources may also change between RLC retransmissions. To handle these variations, already transmitted PDUs can be (re)segmented for retransmissions. The reordering and retransmission mechanisms described in the preceding paragraphs still apply; a PDU is assumed to be received when 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.