3 Radio transmission and reception

This chapter describes the techniques that are used for radio transmission and reception between the mobile and the radio access network. The first section reviews the use of wideband code division multiple access for transmission and reception in release 99. It concentrates on the air interface's physical layer, which is where most of the important processes take place, but it also notes the procedures used in higher layers. We then describe a technique known as high speed packet access, which has been progressively introduced from release 5 with the aim of increasing the rate at which data can be transferred. Finally, we discuss the performance of UMTS, by noting the peak and average data rates that can be achieved, and the advantages and disadvantages that CDMA has compared with other multiple access techniques.

The material in this chapter is more technical than that in later chapters of the book. However, it is unnecessary to take it all in on a first reading, as most of the chapter is self-contained. Instead, a basic understanding of Section 3.1 will be enough for Chapters 4, 5 and 6.

3.1 Radio transmission and reception in release 99

In this first section, we will describe the techniques used for radio transmission and reception in release 99. This is probably the most important part of the system: it is crucial for the delivery of high data rates to the user, and it is the part that has changed the most since the days of GSM.

The action takes place in the air interface's transport protocols, which are illustrated in Figure 3.1. In the transmitter, the radio link control and medium access control protocols handle tasks such as retransmissions and control of the transmitted data rate. The physical layer then manipulates the data in three stages. In the first stage, the data are processed one bit at a time, to carry out tasks such as error correction coding. In the second stage,

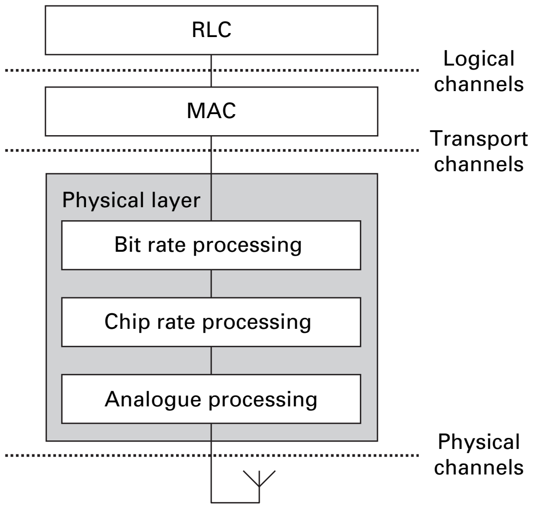

Figure 3.1 Architecture of the air interface's transport protocols, showing the internal architecture of the physical layer.

the coded bits are divided into shorter units called *chips*, and the chips are processed one at a time using the techniques of CDMA. Finally, the chips are converted from digital to analogue form for transmission over the air interface.

As an example, we can illustrate the division of bits into chips by describing how the mobile might transmit a voice call. At the point where the data enter the physical layer, the data rate is typically 12.2 kbps. Using error correction coding and another process called rate matching, the bit rate processor increases the bit rate by a factor between 2 and 3, here to 30 kbps. The chip rate processor then divides each coded bit into 128 chips, to produce a chip rate of 3.84 million chips per second (Mcps). The same chip rate is used throughout UMTS FDD mode, but the other numbers can vary from one data stream to another, and between the uplink and downlink.

We will start by discussing the chip rate processor, which is where most of the new features of UMTS are located. After a detour to the analogue processor, we will move upwards through the protocol stack and finish at the RLC.

3.1.1 Principles of CDMA in UMTS

In this section, we will describe the basic principles of CDMA. The treatment starts with the downlink, using a greatly simplified network that contains just one mobile and one cell. We then develop things to include multiple mobiles and multiple cells, and finally look at the differences between the downlink and the uplink. The description is somewhat simplified: we will note a few differences in the actual implementation in the sections that follow.

Figure 3.2 shows the chip rate processing that is carried out on the downlink, in a network containing one mobile and one cell. At the top of the figure, the base station's chip rate transmitter is handling a stream of bits that it wishes to send to the mobile. The base station assigns the mobile a code that is known either as a *channelisation code* or a *spreading code*: this is made of chips and has a length equal to the

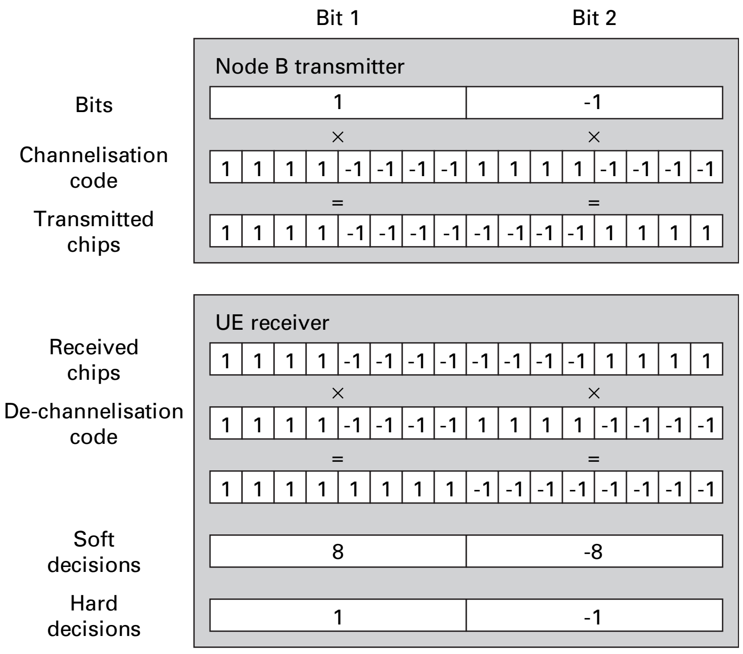

Figure 3.2 Simplified illustration of downlink channelisation and de-channelisation, from a base station to a single mobile.

bit duration, so that the code is repeated every bit. It then multiplies the symbol representations of the bits and chips together, and sends the resulting chips to the analogue processor for transmission.

The bits and chips both have values of 0 and 1, but we have represented them using binary phase shift keyed (BPSK) symbols of +1 and –1. We will continue to use this symbol representation in the discussion that follows, but we will normally describe the quantities as bits and chips rather than symbols, to make it easier to distinguish the bits and chips from each other.

In UMTS FDD mode, the chip rate is fixed at 3.84 Mcps, so the chip duration is about 0.26 μs. The number of chips per bit is called the *spreading factor*: in this example, the spreading factor is 8. The bit rate equals the chip rate divided by the spreading factor, so here the bit rate is 480 kbps. (Note that error correction has already been applied to these bits, so the underlying information rate is typically one third to one half as great.)

If we ignore problems like noise and propagation loss, then the mobile's chip rate processor receives an exact replica of the transmitted chips. We now assume that the base station has previously told the mobile about the channelisation code that it will use, so that the mobile can use this information to undo the effect of channelisation. It does this by multiplying the incoming chips by the channelisation code (or, more accurately, by multiplying their symbol representations together).

The mobile now has to convert the chips into bits, which it does by adding together the chips that comprise each bit. The result is a set of *soft decisions*, each of which has a sign corresponding to the mobile's best estimate of the transmitted bit, and a magnitude corresponding to the mobile's confidence in that estimate. Finally, the mobile converts the soft decisions into *hard decisions* by taking the sign. We can see that the mobile has recovered the original bits: it has done this because the process of channelisation was reversed in the mobile's de-channelisation stage.

Figure 3.3 shows what happens if there are two mobiles in the cell. Here, the base station assigns a different channelisation code to the second mobile, with the condition that the two codes must be *orthogonal*: if we

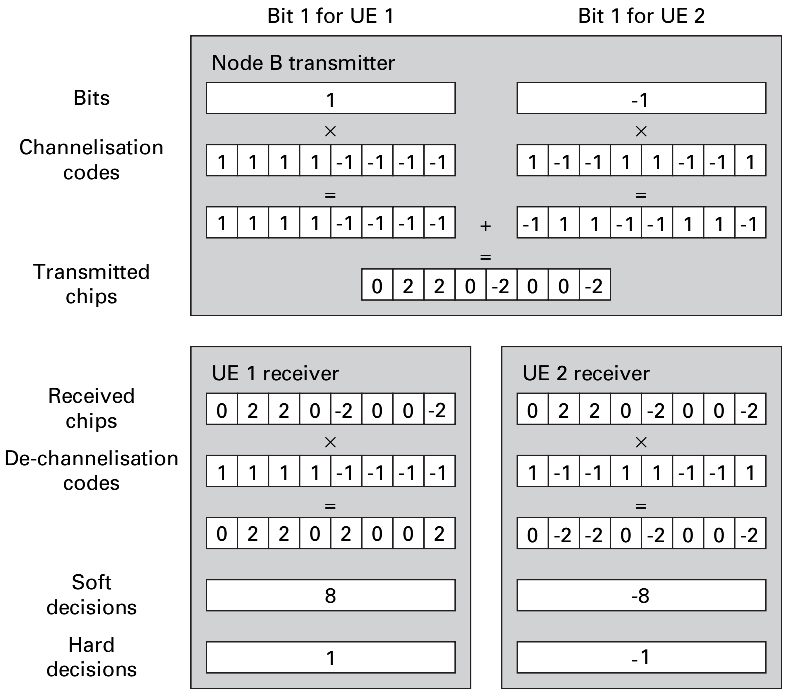

Figure 3.3 Simplified illustration of downlink channelisation and de-channelisation, from a base station to two mobiles.

multiply them together chip-by-chip and add up the results, the total must be zero. The base station multiplies the incoming bits by their respective channelisation codes as before, and then adds the two streams together, chip-by-chip. The transmitted chip stream contains signal levels of +2, 0 and –2, where each chip has contributions for both of the two mobiles. The receive processing is unchanged: each mobile multiplies the incoming stream of chips by its own channelisation code, adds together the chips that comprise each bit, and calculates a set of hard decisions.

In the figure, the two mobiles have successfully recovered the bits that were intended for them, despite the fact that the transmitted stream contained information for both mobiles. This works because the two channelisation codes are orthogonal. Let’s consider, for example, what happens to the symbol that was intended for mobile 1, when it is received by mobile 2. The symbol was transmitted using mobile 1's channelisation code, so when it is processed using mobile 2's de-channelisation code, the result is zero. It therefore has no effect on the soft decisions that are made by mobile 2.

The number of orthogonal codes available is equal to the spreading factor: eight orthogonal codes at a spreading factor of eight, for example. Figure 3.4 shows the codes that are actually used by UMTS. We can think of the figure as a family tree in which each channelisation code has two children, one made by repeating it, and the other by repetition and inversion. The codes on each spreading factor are all mutually orthogonal, while codes on different spreading factors are orthogonal too, so long as they are not ancestors or descendants of each other. The spreading factors are implicitly restricted to integer powers of 2: in release 99, we only use spreading factors from 4 to 512 on the downlink, and 4 to 256 on the uplink.

What happens if there is more than one cell? The channelisation code tree can only accommodate a limited number of mobiles, so we want to re-use it in every cell. This causes a problem if two nearby cells are transmitting on the same frequency and the same channelisation code, because of cross-talk between the two transmissions. We solve the problem by introducing a second set of codes, known as *scrambling codes*, and

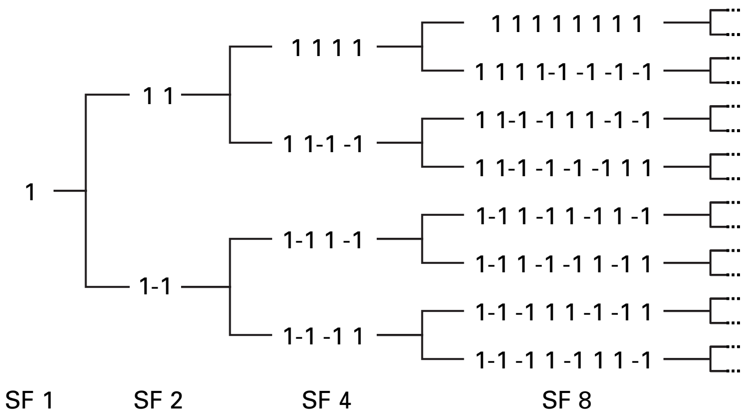

Figure 3.4 Channelisation codes used by UMTS. (Adapted from 3GPP TS 25.213.)

labelling each nearby cell with a different scrambling code. The scrambling codes are made of chips, but they have a much longer repetition period than the channelisation codes: 10 ms, which is 38 400 chips. In UMTS, there are enough scrambling codes to label 512 different cells. This number is large enough that cells on the same scrambling code are a large distance apart, and the cross-talk between them is minimal.

Ideally, the scrambling codes would be orthogonal to each other, but we have unfortunately used up all the orthogonal codes in making the tree of channelisation codes. Instead, we do the next best thing: we create them using a pseudo-random number generator, which makes the scrambling codes *uncorrelated*. If we multiply two scrambling codes together and add up the results as before, then the total is zero on the average, but it is not identically zero.

The effect of using uncorrelated codes is shown in Figure 3.5, which has four cells with different scrambling codes, each transmitting to a

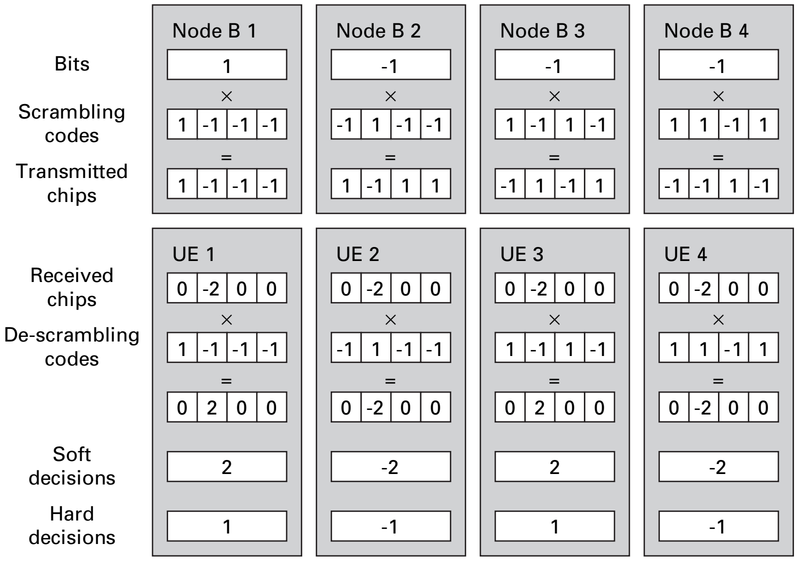

Figure 3.5 Simplified illustration of scrambling and de-scrambling on the downlink. Each mobile receives a signal from its corresponding Node B, and interference from the other three. The processes of channelisation and de-channelisation are omitted.

different mobile. In this figure, the spreading factor is four, so we only show the first four chips from each scrambling code. The channelisation codes are left out for clarity: we could choose channelisation codes of 1111 throughout, which would leave all the other numbers unchanged. We also assume that each mobile receives equally strong signals from the four cells; this is a rather artificial situation, but it serves to illustrate the point.

Each cell applies its scrambling codes by a chip-by-chip multiplication, and the process is reversed in the mobile receiver. Because the scrambling codes are uncorrelated and not orthogonal, the mobiles receive some interference from neighbouring cells. This perturbs the soft decisions away from their expected values, and occasionally (as in the case of mobile 3) causes errors in the hard decisions. The receiver can correct most of these errors later on using error correction and retransmissions, but some of them will leak through and degrade the performance of the application. As we will discuss in Section 3.3, the interference and the resultant errors are a very important issue in UMTS, and will ultimately limit the capacity of the system.

In the uplink, the processing steps are exactly the same, but the channelisation and scrambling codes are used differently. The reason is that, in FDD mode, the transmissions from different mobiles are not time synchronised in any way. This simplifies the design of the system, but it has a disadvantage: it is impossible to distinguish the mobiles by the use of different channelisation codes, because those codes are only orthogonal if they are time synchronised with each other. Instead, the network distinguishes different mobiles by assigning different scrambling codes to them, whichever cell they are in. (The total number of uplink scrambling codes is about four million.) The channelisation codes are only used for two purposes: to set the data rate by means of the spreading factor, and to distinguish different transmissions from a single mobile.

We will discuss the exact implementation of the chip rate transmitter and receiver in Sections 3.1.3 and 3.1.5 below, with the underlying analogue processing coming between them in Section 3.1.4. First, however, we need to digress to an issue that is important for chip rate processing in CDMA: power control.

3.1.2 Power control

Power control is best introduced by describing a problem on the uplink known as the *near far effect*. Let's assume that a base station is receiving signals from two mobiles, one close to the base station and one far away. The base station's analogue-to-digital (A/D) converter has a limited dynamic range (just a few bits), so if the two signals are transmitted with the same power, then the nearby one can saturate the A/D converter while the distant one is hardly resolved at all. The base station will then be unable to hear the distant mobile. This is not a problem for systems based on FDMA or TDMA, where the signals can be separated before digitisation using their carrier frequencies or arrival times, but it is an important issue for CDMA.

The solution depends on the physical channel of interest. For the dedicated physical data channel (DPDCH) and the dedicated physical control channel (DPCCH), the answer lies in fast closed loop power control (Figure 3.6). Every two-thirds of a millisecond, the base station measures the *signal-to-interference ratio* (SIR) that it receives from each

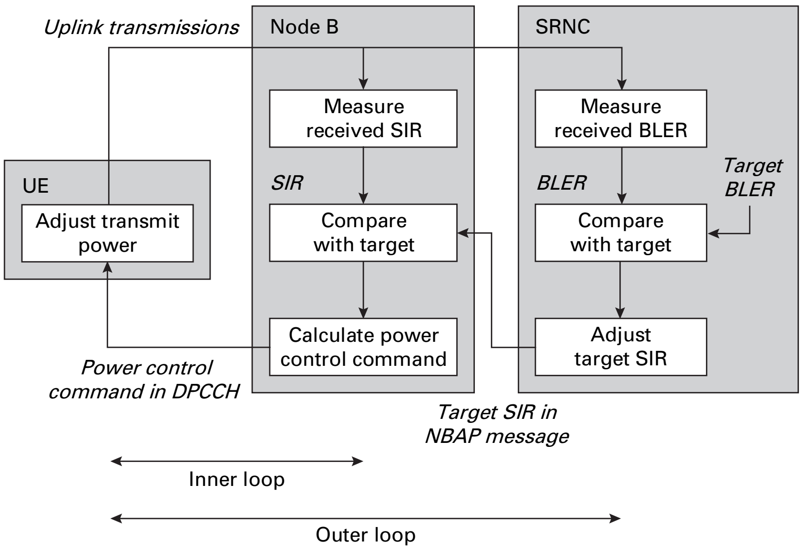

Figure 3.6 Operation of fast closed loop power control for the DPDCH and DPCCH uplink.

mobile, and compares it with a target SIR. It then uses the result to compute a power control command, which it sends to the mobile on the downlink DPCCH.

The power control command is calculated as follows. If the received SIR is below the target, then the base station tells the mobile to increase its transmit power, typically by 1 dB. Similarly, if the SIR is above the target, then the base station tells the mobile to decrease its transmit power. As a result, the mobile adjusts its transmit power up or down every two-thirds of a millisecond, so as to hold the base station's received SIR close to the target value.

The target signal-to-interference ratio is itself adjusted using an outer loop, as follows. Over a longer timescale, the serving RNC measures the error rate in the received signal using a quantity called the *block error ratio* (BLER). It then compares this with a target BLER that is associated with the data stream. (We will define the BLER in Section 3.1.6, and list some suitable target values.) If the BLER is greater than its target value, then the serving RNC sends a signalling message to the Node B, in which it tells the Node B to adjust the target SIR upwards. The Node B uses the target SIR in its inner loop, so it immediately tells the mobile to increase its transmit power. This causes the error rate at the serving RNC to fall.

If the BLER is less than the target value, then the serving RNC tells the Node B to adjust its target SIR downwards, and the Node B tells the mobile to reduce its transmit power. This clearly increases the number of errors in the received data stream, but it brings two important benefits. First, by reducing the mobile's transmit power, we increase its battery life. Second, we reduce the amount of interference at the base station receiver, so we increase the capacity of the cell. We can sum up the effect of the control loop by saying that the mobile's transmissions are at the lowest power consistent with satisfactory reception of the signal.

Fast power control brings one more advantage, which is shown in Figure 3.7. If a mobile moves into a fade, then the base station tells it to increase its transmit power, so as to keep the received signal-to-interference ratio at roughly the same level. This reduces the amount of fading in the received signal, so it reduces the error rate. As shown in

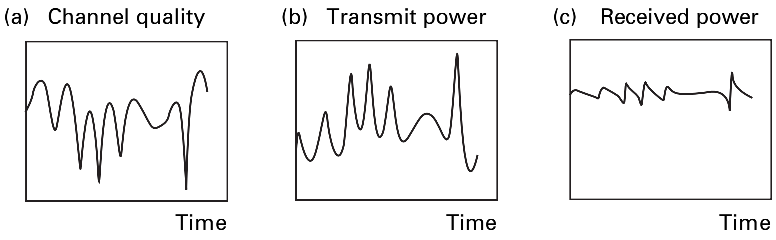

Figure 3.7 Illustration of how fading can be reduced by the use of fast power control. (a) Received signal-to-interference ratio in the absence of fast power control. (b) Transmitted power when using fast power control. (c) Received signal-to-interference ratio in the presence of fast power control.

the figure, we cannot remove fading altogether because of issues such as time delays in the control loop, but the improvement is an important one nevertheless.

For these reasons, fast power control is used for the downlink DPDCH as well as for the uplink. On the downlink, fast power control is applied independently to every data stream, so the base station sends high power signals to mobiles that are far away, and low power signals to mobiles that are nearby.

Power control is also important for the physical random access channel (PRACH). The mobile typically uses this channel for short transmissions such as small bursts of packet data or discrete signalling messages. It is harder to control the power of the PRACH, because the mobile has to work out its transmit power without the help of a continuously running control loop. The solution is shown in Figure 3.8. The mobile starts by sending a preamble made of short bursts. The first burst is at a low enough power that the base station is unlikely to hear it, while each subsequent burst is typically 2 or 3 dB stronger than the previous one. Once the base station hears the mobile, it sends a reply on the acquisition indicator channel (AICH). The mobile can then send its uplink data, confident that its transmit power is strong enough for the base station to hear it, but not so strong as to cause unnecessary interference to the base station. The messages are very short (either 10 or 20 ms long), so we can assume that the required transmit power will not change significantly before the end of the message. The mobile therefore transmits the message with a constant power.

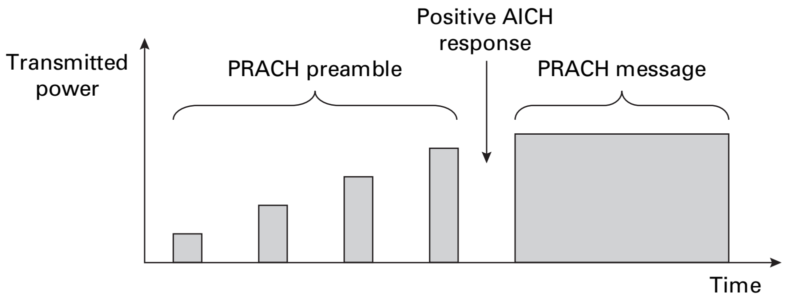

Figure 3.8 Operation of open loop power control for the PRACH.

The remaining physical channels are downlink-only channels that are designed to cover the whole of the cell. They are usually sent at a constant power, without any further adjustment.

3.1.3 Chip rate transmitter

In this section, we will look at how channelisation and scrambling are actually implemented in UMTS. We will just do this for the DPDCH and DPCCH: the other physical channels are broadly similar, but have a number of differences in the detail.

Figure 3.9 shows how the two channels are organised, at the point where they enter the chip rate transmitter. In both the uplink and downlink, the information is laid out in 10 ms *frames*, which are divided into 15 *slots* of $^2/_3$ ms each. Within each slot, the DPDCH and DPCCH are sent in parallel on the uplink but in series on the downlink. The DPDCH contains data, while the DPCCH has four different control fields. The *pilot* information is a known stream of bits, which help the rake receiver (Section 3.1.5) to process the incoming data stream. The *transport format combination indicator* (TFCI) signals the type of information that the DPDCH frame contains, such as voice, signalling or both. The *transmit power control* (TPC) bits are the power control commands described above. The *feedback information* (FBI) bits are less important than the others: they are only used on the uplink, and control an optional process called closed loop downlink transmit diversity.

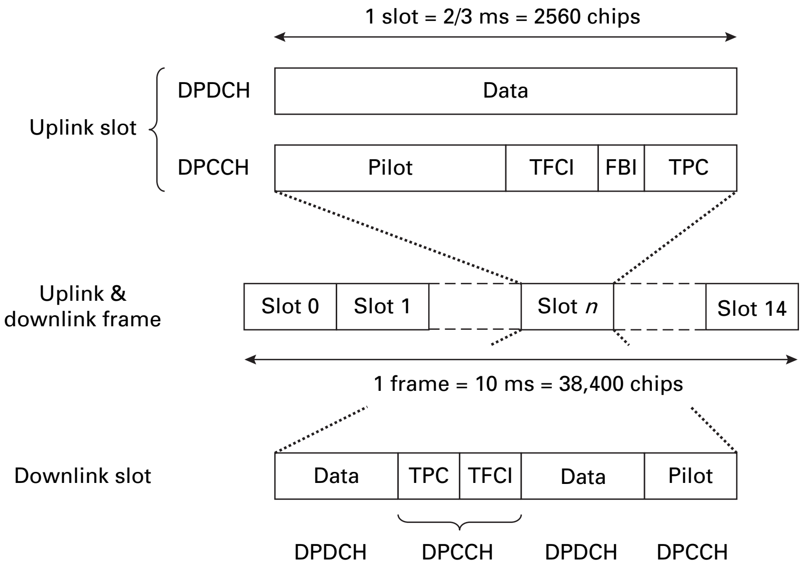

Figure 3.9 Frame and slot structure for the DPDCH and DPCCH. (Adapted from 3GPP TS 23.211.)

Figure 3.10 is a block diagram of the uplink transmitter. As shown in the figure, the uplink normally uses one DPDCH and one DPCCH. If the mobile is sending more than one information stream (such as voice and signalling, for example), then the streams are normally multiplexed onto the same DPDCH, with the TFCI indicating the type of information being sent. The mobile converts the bits into BPSK symbols, and multiplies them by a channelisation code. The DPDCH uses a spreading factor that depends on the bit rate, while the DPCCH uses a fixed spreading factor of 256. The specifications define the exact choice of channelisation codes.

The mobile then determines how much power is transmitted in the two channels, by multiplying the data by suitable scaling factors. To transmit a high power signal, for example, it might convert symbols of ±1 into ±10, while for a low power signal it might convert them into ±0.1. These scaling factors are actually applied in two steps. The first step comes after channelisation, where the mobile applies digital scaling factors to set the relative transmit powers of the two channels. The second step comes in the analogue transmitter, which can handle a much bigger dynamic range.

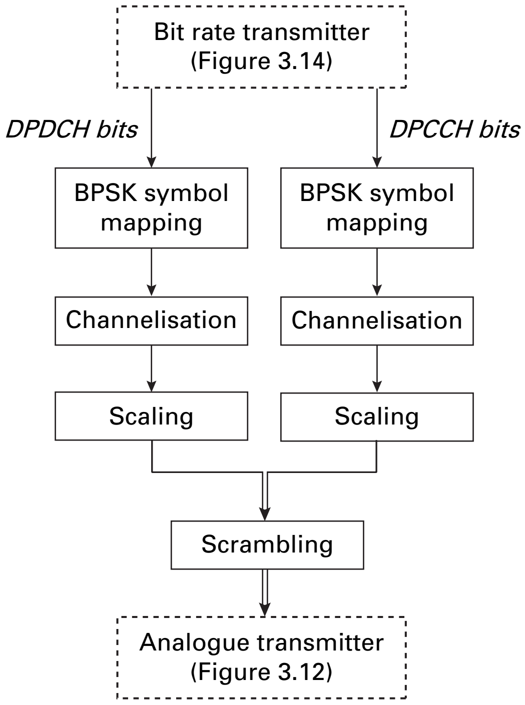

Figure 3.10 Block diagram of the uplink chip rate transmitter for the DPDCH and DPCCH, in the usual case of transmission on a single DPDCH.

There, the mobile adjusts the absolute transmit power using the power control commands received from the Node B.

Between these two steps is the scrambling operation, which is slightly more sophisticated than the process described earlier but works on the same basic principles. The DPDCH and DPCCH are treated as the real and imaginary parts of a complex number, otherwise known as the in-phase and quadrature components (I and Q). The complex number is indicated in the figure by a double line. The scrambling code is a complex number as well, and the scrambling operation includes a complex multiply.

This process works fine at information rates up to about 384 kbps, but at higher rates more data channels are required. In release 99, a mobile can use up to six uplink data channels with a spreading factor of four, to give a maximum data rate of about 2048 kbps. Three of the channels are transmitted on the in-phase component and three on the quadrature component, which ensures that the data streams are still orthogonal to each other, even though each channelisation code is used twice. In practice, it is very unusual for mobiles to transmit at such high data rates in this way. Instead, they normally use the techniques of high speed uplink packet access that are described in Section 3.2.

The downlink (Figure 3.11) is handled a bit differently. At the start of the process, the bits for each mobile are taken two at a time and converted into quadrature phase shift keyed (QPSK) symbols, which are treated as complex numbers with values of $\pm 1 \pm i$. After channelisation, the symbols are scaled using the power control commands received from their respective mobiles. The symbols for different mobiles are then added together, scrambled, and sent to the analogue transmitter as before. The maximum data rate on the downlink is again about 2048 kbps per mobile.

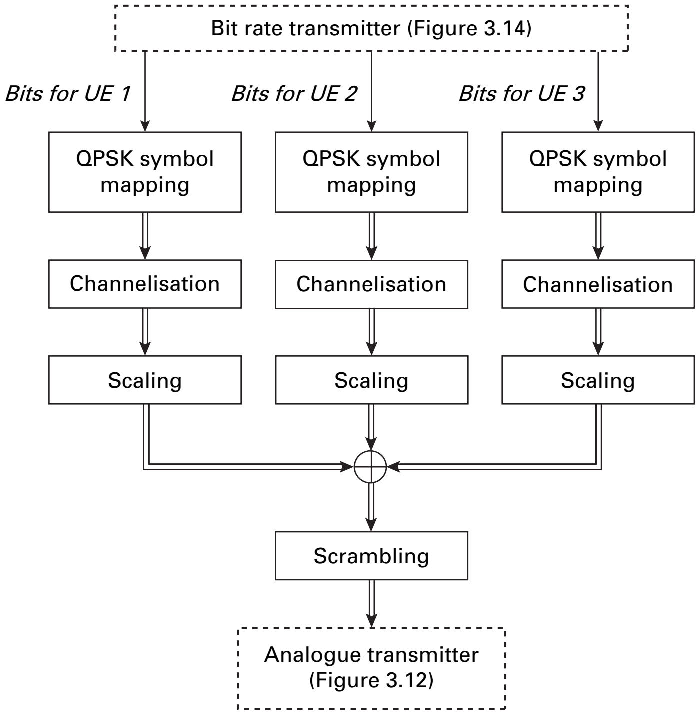

Figure 3.11 Block diagram of the downlink chip rate transmitter for the DPDCH and DPCCH.

The reason for the difference between the uplink and the downlink is as follows. On the downlink, the channelisation codes are a scarce resource. The DPDCH and DPCCH are therefore transmitted in series, on the same channelisation code, which uses the codes in the most efficient way. This technique has one disadvantage, however: if the two channels require a different transmit power, then serial transmission makes the transmitted power vary at a frequency of $(^2/_3\,\text{ms})^{-1}$, which is 1.5 kHz. This is an audio frequency, so it causes interference to radios and other audio devices. On the uplink, we can avoid any risk of interference by transmitting the DPDCH and DPCCH in parallel, to keep the transmit power constant. On the downlink, interference is usually much less of a problem for two reasons: the transmitter is further from any susceptible devices, and the transmissions to multiple mobiles keep the total power much the same.

3.1.4 Analogue processing

There are various ways of implementing the analogue processor. In this section, we will just give an outline of one such implementation, to highlight the principles that are involved.

Figure 3.12 shows an analogue processor that uses direct conversion in both the transmitter and the receiver, with no intermediate frequency. The I and Q streams are first passed through a particular type of digital filter known as a *Nyquist filter*, which smoothes the transitions between successive chips, without disturbing the waveform at the sampling times the receiver will eventually use. The effect of the filter is to reduce the signal bandwidth in the frequency domain, which ensures that the signal can be transmitted and received in the available bandwidth of 5 MHz. The Nyquist filter used in UMTS is a *raised cosine* filter, but it is actually implemented as two separate *root raised cosine* filters. One of these is in the transmitter to reduce the transmitted signal bandwidth, and the other is in the receiver to reduce interference from signals on nearby carriers.

In the transmitter, the filtered I and Q streams are converted from digital to analogue, and are amplified and filtered at baseband. They are then converted to radio frequency (RF) by mixing them with cosine and sine wave carriers, and added together. The signal is then amplified and

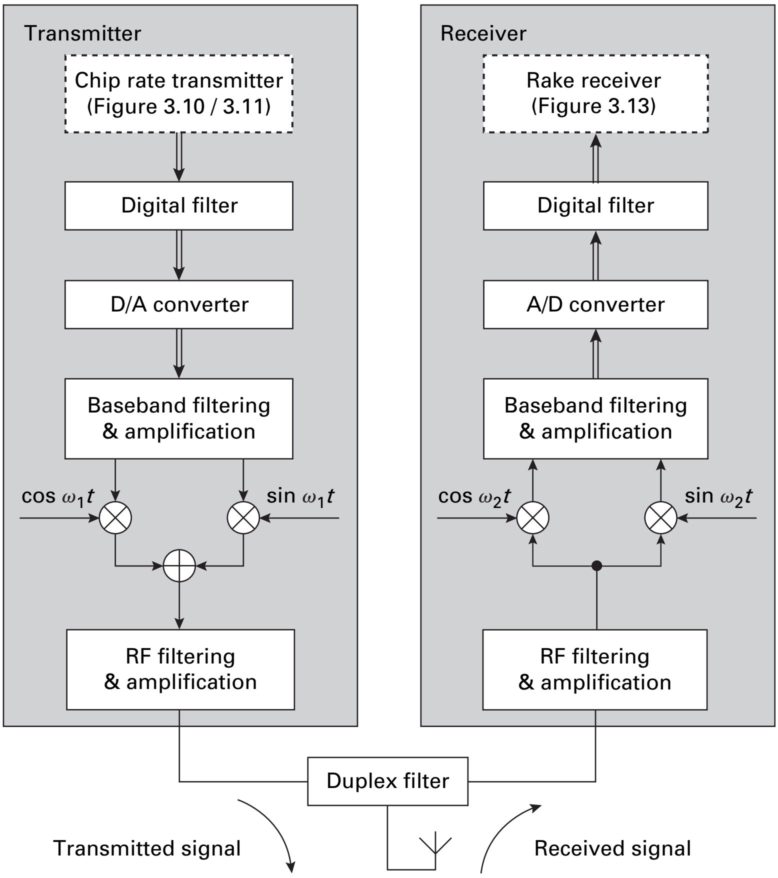
Transmitter
Chip rate transmitter (Figure 3.10 / 3.11)
Digital filter
D/A converter
Baseband filtering & amplification
cos $\omega_1 t$
sin $\omega_1 t$
RF filtering & amplification
Receiver
Rake receiver (Figure 3.13)
Digital filter
A/D converter
Baseband filtering & amplification
cos $\omega_2 t$
sin $\omega_2 t$
RF filtering & amplification
Duplex filter
Transmitted signal
Received signal

Figure 3.12 Implementation of the analogue processor using direct conversion in both transmitter and receiver.

filtered at RF before being sent to the antenna, by way of a duplex filter that isolates the transmit and receive chains from each other.

In the receiver, the processing is reversed. The incoming signal is amplified and filtered at RF, before being mixed with cosine and sine wave carriers that convert it to baseband. After low pass filtering and a further stage of amplification, the baseband signals are digitised and passed to the root raised cosine filter.

Before we end this section, it is worth noting a problem that affects the receiver's processing: the phase angle of the received signal is at this stage completely unknown. If, for example, we move the receive antenna through half a wavelength of the carrier (typically 75 mm in UMTS), then the received waveform is inverted, and we change the signs of all the chips received on I and Q. Even worse, if we move the antenna through a quarter of a wavelength, then (ignoring a sign change) we interchange I and Q. Generalising, we can say that the I and Q signals in the receiver are not the same as the ones in the transmitter, but instead are linear combinations of them:

$$\begin{aligned} I_{\text{received}} &= I_{\text{transmitted}} \cos\phi + Q_{\text{transmitted}} \sin\phi \\ Q_{\text{received}} &= Q_{\text{transmitted}} \cos\phi - I_{\text{transmitted}} \sin\phi \end{aligned} \tag{3.1}$$

for some phase shift ϕ. The situation is worse in a multipath environment, where there are several copies of the received signal, each of which has a different value of ϕ. If we are to process the received signal successfully, then we have to measure and remove those phase shifts, so as to map the received chip streams onto the ones that were originally sent. This will happen as part of the rake receiver processing, which we will now discuss.

3.1.5 Rake receiver

The receiver's chip rate processor uses a device known as the *rake receiver*, which is shown in Figure 3.13. We will describe it by referring to the downlink, but the treatment refers equally well to the uplink.

The aim of the receiver is to reduce the amount of fading in a multipath environment, by applying diversity processing to the incoming rays. We therefore assume that the mobile is receiving two distinct rays from the base station. In the receiver, we set up two processing streams that are known as *fingers*: the two fingers receive exactly the same information, but fingers 1 and 2 are configured to process the information arriving on rays 1 and 2 respectively.

At the start of finger 1 is a delay buffer, which is configured so that the mobile's de-scrambling code is time aligned with the chips that are arriving on ray 1. This implies that, at the output of the de-scrambling stage, finger 1 contains a de-scrambled signal from ray 1. However,

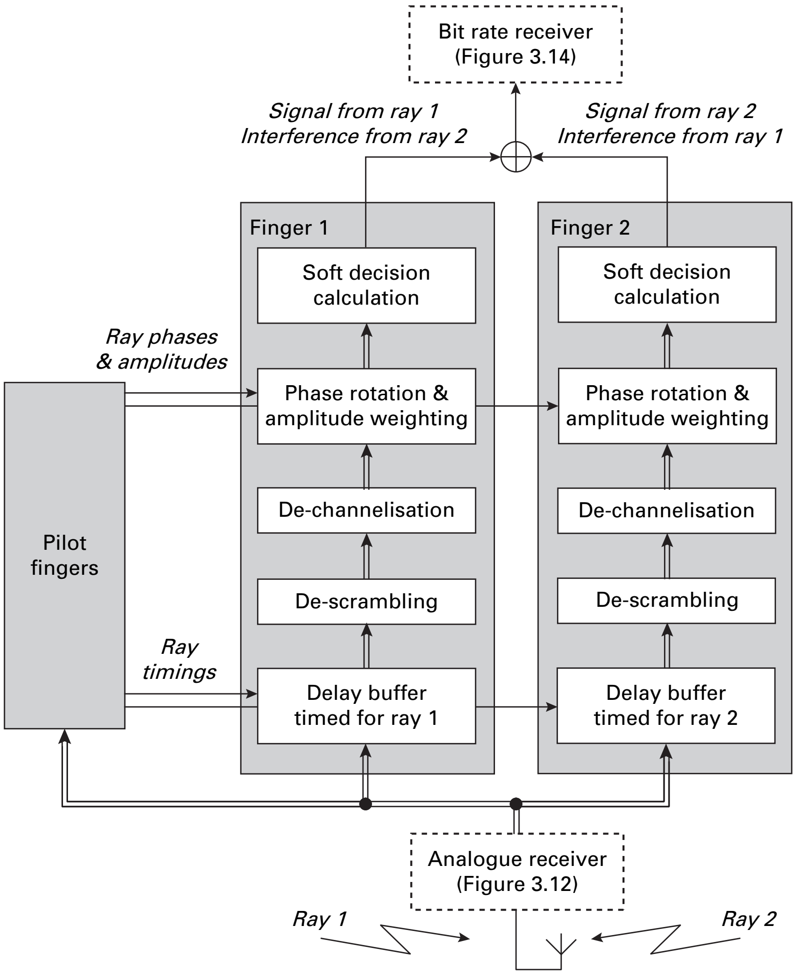

Figure 3.13 Block diagram of the rake receiver, for the case where the receiver is processing two distinct rays.

finger 1 is also receiving chips that arrive on ray 2. These have a different arrival time, so they are misaligned with the mobile's de-scrambling code, and they generate a small amount of interference. The processing in finger 2 is the same, except that its delay buffer is configured to align the de-scrambling code with the chips arriving on ray 2. The effect is that finger 2 produces a signal from ray 2 and interference from ray 1.

After de-scrambling, the data are de-channelised and converted to soft decisions as before. The calculation of hard decisions is left until much later, in the bit rate receiver.

We also need to remove the phase shifts ϕ that we noted above. To do this, the receiver processes the pilot bits on either the DPCCH or the downlink's common pilot channel (CPICH), using separate rake fingers that are configured using the channelisation code for the corresponding channel. The transmitted pilot bits are defined in the 3GPP specifications, and are received with the same phase shift as all the other data. By comparing the received pilot bits with the expected ones, we can measure the value of ϕ for each ray and send the measurement to the corresponding rake finger, which removes the phase shift from the data stream. This process allows us to add together the two sets of soft decisions at the end of the rake receiver, without any risk of destructive interference between them. We also apply a weighting factor to each data stream, which is computed using the amplitude of the corresponding pilot signal. The weighting factor ensures that the signal-to-interference ratio, after the two data streams are added together, is as large as possible.

The receiver has a couple of other tasks, which are not shown in the figure. First, a component known as the *searcher* runs the cell search procedure that we will describe in Chapter 4, so as to discover when new rays appear and old rays disappear. The searcher sends information about the rays it finds to the rest of the receiver, which responds by setting up new rake and pilot fingers, and tearing down the ones that are no longer required. Second, there are extra components associated with the pilot fingers that measure the arrival time of each ray, by locking on to the expected sequence of pilot bits and adjusting for any timing changes.

The effect of the rake receiver is that we have processed the two rays independently, in much the same way as for receive antenna diversity, so as to reduce the amount of fading in the received signal. However, the process only works if the difference in the arrival times of the two rays is greater than the chip duration. (If they arrive closer together, then the rake receiver cannot distinguish them, so they are handled by a single finger and cause fading as before.) This implies that the path difference between the rays must be greater than about 80 m, which is the distance travelled by a radio signal over the duration of one chip. This is easy to achieve in an urban macrocell that contains lots of widely spaced reflectors, but it is harder in a rural environment or a microcell, and is usually impossible in an indoor picocell.

If there are several distinct rays, then the receiver sets up one finger for each ray, starting with the highest power ray and moving on to weaker ones. Two issues limit the number of fingers used: the processing power in the receiver, and the diminishing returns that result from processing weaker and weaker rays. If there are several fingers, then they start to look like the prongs on a garden rake, which explains the receiver's name.

Finally, note that the receiver does not care where the two rays in Figure 3.13 come from. On the downlink, for example, the rays can come from different cells, so long as the mobile uses different de-channelisation and de-scrambling codes in the two fingers. This means that a mobile in soft handover can process the rays from different cells by adding them together, in the same way that it processes rays from a single cell. As a result, most of the processing needed for soft handover is just a by-product of the processing in the rake receiver.

3.1.6 Bit rate processing

We will now move upwards through the physical layer and look at the bit rate processor. Figure 3.14 summarises the processing that it carries out: there are more blocks in practice and the details are different on the uplink and downlink, but the figure covers all the important issues.

In the transmitter, data arrive on the transport channels, in the form of physical layer service data units that are known as *transport blocks*. These have a duration known as the *transmission time interval* (TTI), which can take a value from 10 to 80 ms (1 to 8 frames) depending on the particular data stream. Roughly speaking, each transport channel carries a single stream of information such as voice, signalling or packet data.

Skipping over the first process for the moment, the second transmit process is error correction coding, which we introduced in Chapter 1. In this process, we represent the information bits using codewords, so as to double or treble the number of transmitted bits. The algorithm used is

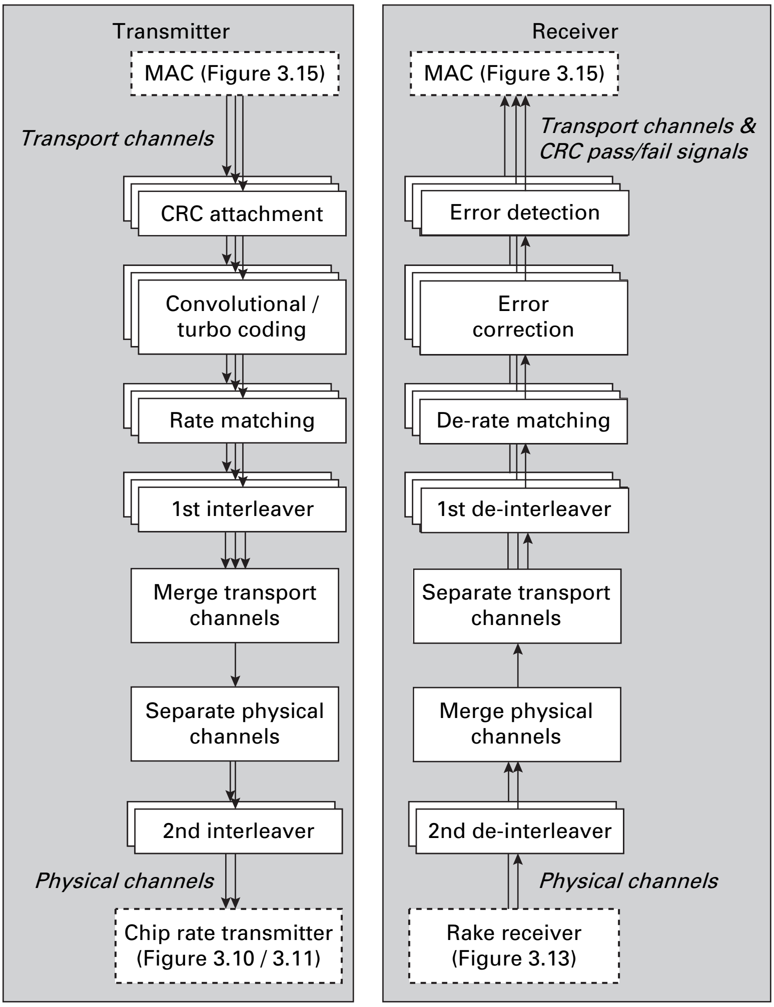

Figure 3.14 Simplified block diagram of the bit rate processor. (Adapted from 3GPP TS 25.212.)

either *convolutional coding* or *turbo coding*: the first is used at low bit rates, while the second gives better results at bit rates above about 32 kbps. The receiver uses the extra information for error correction, which greatly reduces the error rate. The receiver's hard decisions are usually computed as part of the error correction stage, so the soft decisions run as far as the input to it.

Unfortunately there is no such thing as a perfect error correction algorithm, so a few errors will leak through. These are handled by the blocks we skipped over: *cyclic redundancy check* (CRC) *attachment* in the transmitter, and *error detection* in the receiver. In the transmitter, we use the bits in the transport block to compute and append a few extra bits that are known as the CRC. In the receiver, we examine the transport block and CRC to see if they are still consistent. If they are, then everything is fine; if not, an error has occurred. The action we then take depends on the nature of the data stream. For real time streams such as voice, timely delivery is more important than accuracy, so we just tell the application software about the error and let it decide what to do. (It might estimate the erroneous block's contents using the previous block, for example.) For non-real time streams such as emails or web pages, we send a signal to the RLC protocol, which asks for a retransmission.

The percentage of transport blocks that fail the CRC is the block error ratio (BLER) that we introduced earlier. Voice and video streams run happily with BLERs up to around 1% and 3% respectively, and these are typical target BLERs for those streams. For packet data, the target BLER is usually around 10%: we can tolerate such a high figure because the errors are corrected later on using retransmissions.

Like the error correction stage, error detection is not perfect: a transport block can occasionally pass the CRC, even though it contains bit errors. These residual bit errors leak through to the application, and in some cases (emails and web browsing, for example) have the potential to be a serious problem. By using enough CRC bits, the network can reduce the residual bit error ratio to a level the application can tolerate.

Rate matching deals with a problem that is implicit in our description so far: the transmitter has to handle any bit rate that the application throws at it, but the physical layer only uses spreading factors that are an integer power of 2. If the input stream contains too many bits for the target spreading factor, then the transmitter removes a few of the bits by a process called *puncturing*. For example, the error correction algorithm might represent a set of 4 coded bits using a 12-bit codeword. The puncturing algorithm might match the chosen spreading factor by removing one of these, reducing the length of the codeword to 11 bits. The receiver runs the same algorithm, so it can work out the places where the punctured bits occurred: it then inserts soft decisions of 0 at the appropriate places, and leaves the error correction stage to recover the information bits. If the transmitter has too few bits, then it either duplicates some of them by *repetition*, or leaves gaps in the transmitted data stream.

Interleaving solves another problem. The coding stage distributed the information from a single input bit over several coded bits, but the coded bits can all be ruined if they arrive within a single fade. Interleaving redistributes the coded bits across the transport block, to ensure that at least some of the coded bits from each information bit arrive correctly. As shown in the figure, interleaving is actually carried out in two stages: the second interleaver distributes the coded bits over a 10 ms frame, and the first interleaver does some extra work if the TTI is longer than a frame.

Between the two interleavers, the transport blocks are broken apart into frames, and the frames from the different transport channels are multiplexed together. If necessary, the information can then be distributed amongst multiple physical channels, although this is usually only done if the data rate is too high for one channel to cope.

3.1.7 Medium access control protocol

Figure 3.15 shows the architecture of the medium access control (MAC) protocol. There are two main components: the MAC-d handles dedicated channels, and the MAC-c/sh handles common channels. The network has one MAC-c/sh per cell, which is implemented in the cell's controlling RNC, and one MAC-d per mobile, implemented in the mobile's serving RNC.

The link between the MAC-d and MAC-c/sh supports the transmission of dedicated logical channels using common transport and physical channels. These are handled as follows. On the downlink, the base station labels the transport blocks using the identity of the target mobile. It then transmits them on the forward access channel (FACH) and the secondary common control physical channel (SCCPCH). When a mobile reads the channel, it processes the transport blocks if the label matches its identity, and discards them otherwise. A similar process takes place on the uplink.

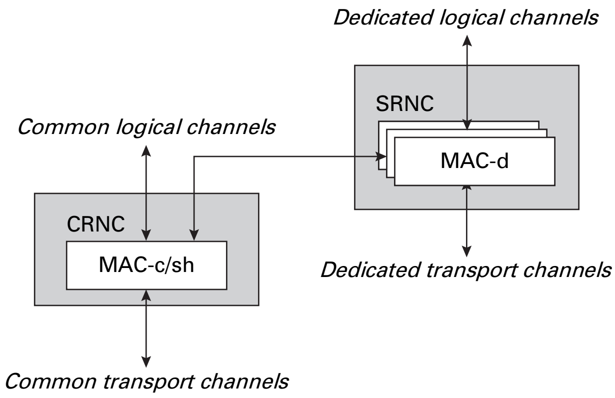
Dedicated logical channels
SRNC
MAC-d
Common logical channels
CRNC
MAC-c/sh
Dedicated transport channels
Common transport channels

Figure 3.15 Architecture of the medium access control protocol. (Adapted from 3GPP TS 25.321.)

The MAC has many functions, but its main role is to decide how many bits are sent per transmission time interval from each of the transport channels. The process works as follows. When the data streams are first set up, the network associates each transport channel with a table known as a *transport format set* (TFS). This lists the allowed transport block sizes for the channel in question, and for each block size, the number of transport blocks that are transmitted in parallel. Each row in the table is labelled using a *transport format indicator* (TFI).

The network also sets up a *transport format combination set* (TFCS). This defines how the transport formats from different channels can be combined, leaving out any combinations which are mutually inconsistent or which would lead to too high a data rate. Each combination is labelled using a *transport format combination indicator* (TFCI). Finally, the network sends this information to the mobile, using separate tables for the channels on the uplink and downlink.

The network also associates each transport channel with a transmission priority. This is set to a high value for a real time stream such as voice, and low for streams that can tolerate delays such as signalling or non-real time packet data.

Every TTI, the MAC finds out how much data are waiting to be transmitted, chooses a row from the transport format combination set, and sends the data to the physical layer. The algorithm in the mobile is well defined: grab as much data as you can from the highest priority channel in a way that is consistent with the transport format combination set, and then move on to the next one. The network has more room for manoeuvre and has to juggle transmissions to all the mobiles in the cell, but would use a broadly similar technique. The MAC also sends the corresponding TFCI to the physical layer, which sends it to the receiving device, for example using the DPCCH (Figure 3.9). This tells the receiver what is going on, so that it can process the incoming data correctly.

Table 3.1 shows an example. Here, the mobile is transmitting two transport channels, both of which have three transport formats (Table 3.1a). Channel 1 adjusts its bit rate by adjusting the transport block size, while channel 2 uses the number of transport blocks. The TFCS could have up to nine transport format combinations, but in this example (Table 3.1b), only six are allowed because the others lead to too high a bit rate. Every TTI, the mobile inspects its transmit buffers, chooses a row from the table and sends the data to the physical layer, along with a note of the selected TFCI.

3.1.8 Radio link control protocol

The last protocol that we will cover is the radio link control protocol. The RLC has three modes of operation, one of which is chosen for each individual data stream. The simplest is *transparent mode* (TM), which passes the transmitted data directly from input to output with little or no further processing. Transparent mode is typically used for voice: the input service data units (SDUs) are the packets produced by the AMR codec, and are mapped directly onto the output protocol data units (PDUs).

Unacknowledged mode (UM) is suitable for real time packet data such as streaming video. The input SDUs are typically IP packets, which are too large to transmit as they are. The RLC therefore cuts them into smaller PDUs, and can splice them together to make a PDU containing the end of one SDU and the beginning of the next one (Figure 3.16). It also adds a

Table 3.1 *Example of a transport format combination set. (a) Transport format sets for two simple transport channels. Each transport format is labelled using a transport format indicator (TFI). (b) Transport format combination set that supports six of the nine possible combinations from the two channels. Each combination is labelled using a transport format combination indicator (TFCI).*

(a) Transport format sets

Channel	TFI	Block	# blocks
1	0	0	1
	1	100	1
	2	200	1
2	0	100	0
	1	100	1
	2	100	2

(b) Transport format combination set

TFCI	TFI for channel 1	TFI for channel 2
0	0	0
1	0	1
2	0	2
3	1	0
4	1	1
5	2	0

header that includes a *sequence number* (SN), so the receiver can notice if any PDUs are missing.

The most complex RLC mode is *acknowledged mode* (AM). This is the only mode that does retransmissions, so it is used for non-real time packet data. It includes all the features of unacknowledged mode, together with

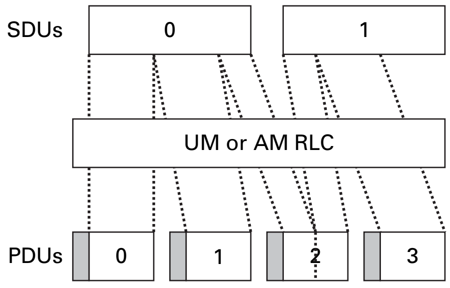

Figure 3.16 Segmentation and concatenation in the unacknowledged and acknowledged mode RLC.

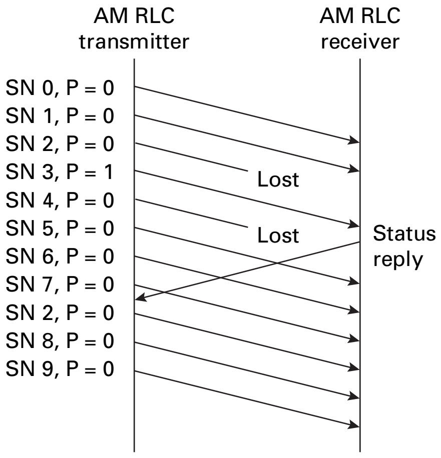

Figure 3.17 Operation of selective retransmission in the acknowledged mode RLC.

the selective retransmission scheme illustrated in Figure 3.17. The transmitter streams packets to the receiver, and also stores them in a buffer in case they need to be retransmitted. Occasionally, it sets a polling bit P in the packet header to request acknowledgements. The receiver replies with status messages that indicate which sequence numbers have passed the cyclic redundancy check and which have failed. The transmitter can then discard the packets that arrived correctly, and retransmit the ones that did not.

When operating in unacknowledged and acknowledged mode, the RLC also encrypts the transmitted data. A quirk of the specifications is that, in transparent mode, encryption is delegated to the MAC.

3.2 High speed packet access

The phrase *high speed packet access* (HSPA) describes a collection of techniques that were introduced in releases 5 to 7. HSPA increases the throughput of the air interface, but it makes the system more complex, and it makes the data rate for each individual mobile more variable. This trade-off means that HSPA is only suitable for streams that can tolerate a variable bit rate, so it is only used for packet switched data streams, not for circuit switched ones.

The techniques were introduced in three stages. *High speed downlink packet access* (HSDPA) was the first implementation to appear, in release 5. Release 6 introduced the *enhanced uplink*, which is more often known as *high speed uplink packet access* (HSUPA). Release 7 brought a number of enhancements, which are collectively known as *HSPA evolution* or *HSPA+*.

The approach we will take is to describe the underlying techniques, and then show how those techniques are used in each of the three implementations. Although HSUPA was introduced second, it is actually closer to the release 99 specifications than HSDPA is. We will therefore cover the implementations in the order HSUPA, HSDPA and finally HSPA+.

3.2.1 Hybrid ARQ with soft combining

The first technique we will cover is *hybrid automatic repeat request* (hybrid ARQ) *with soft combining*. This is used in a similar way in both HSUPA and HSDPA, and mainly affects the bit rate processing in the physical layer.

Hybrid ARQ is the process of correcting errors by the combination of error correction and retransmissions, as described in Sections 3.1.6 and 3.1.8 above. The rationale for adding soft combining to this process comes from thinking about the acknowledged mode RLC. If the receiver's RLC picks up a transport block that has failed the cyclic redundancy check, then it discards it and asks the transmitter for a new one. This works fine, but there is a drawback: the first transport block had some useful signal energy, which has just been lost. If we could find a way to keep the first

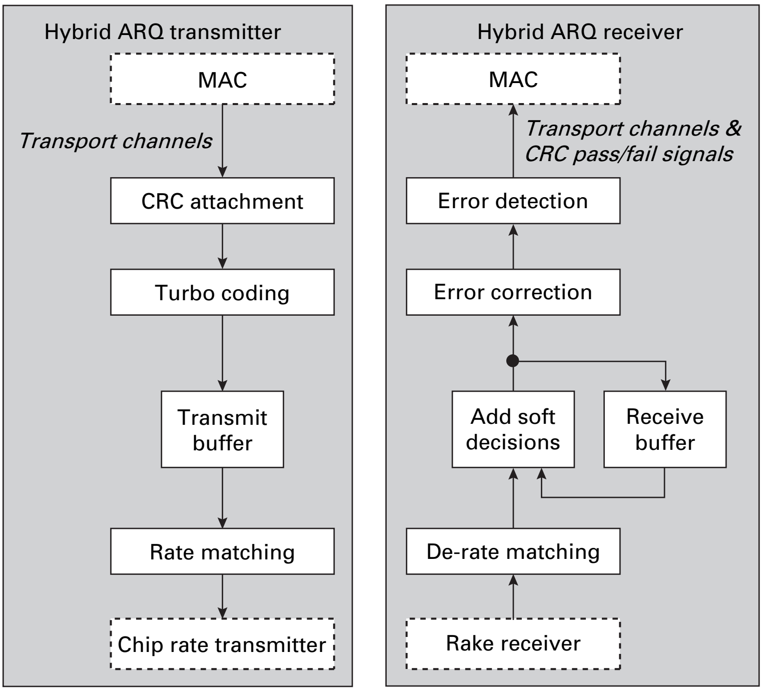

Figure 3.18 Modifications to the bit rate processor when using hybrid ARQ with soft combining.

transport block and combine it with the second one, then we would do better than by using the second one alone.

When using hybrid ARQ together with soft combining, we achieve this by introducing an extra retransmission stage into the physical layer (Figure 3.18). The bit rate transmitter has a buffer after the error correction coder, in which it stores the coded bits in case a retransmission is required. It always uses turbo coding, because high speed packet access is aimed at the high bit rates for which turbo coding is more suitable.

In the receiver, the soft decisions are processed by the lower parts of the physical layer and are stored in a receive buffer, before passing through error correction and error detection as before. Depending on the result of the CRC, the receiver sends a positive or negative acknowledgement back to the transmitter, using a new physical layer signalling message. After a negative acknowledgement, the transmitter sends the transport block again, and the receiver combines the soft decisions from the two transmissions by adding them together. This increases the signal-to-interference ratio, so it increases the probability of the block passing the CRC. (Note that the soft decisions must be added before we reach the error correction stage, as this process is non-linear.)

The process can continue for several retransmissions until eventually we get a CRC pass. One potential problem is that the information in the receive buffer may get corrupted, perhaps by a burst of interference in one of the transmissions. To handle this possibility, the transmitter can give up and move on to the next transport block if it reaches some maximum number of retransmissions. If this happens, then the receiver's RLC notices the CRC failure, and handles the problem in the same way as in release 99.

At a system level, the benefit of this process is that it can use a higher block error ratio than the release 99 scheme, so it can use a lower transmitted signal power. As we will see in Section 3.3, this reduces the interference level in the cell, and increases its capacity. Clearly, however, it requires a data stream that can tolerate the jitter arising from retransmissions.

There is some extra subtlety buried in the rate matching stage. For a retransmission, the transmitter can use either the same puncturing or repetition pattern that it did originally, or a different one. The two cases are sometimes known as single or multiple *redundancy versions*. The processing in the two cases is nearly the same: the only differences are that the receiver needs more storage to handle multiple redundancy versions, and may combine soft decisions that are zero in one transmission and non-zero in another.

Another subtlety comes from the way in which acknowledgements and retransmissions are scheduled. HSPA uses a simpler retransmission scheme than the RLC, known as *stop-and-wait*. Using this technique, the transmitter stops after sending a transport block and waits until it has received an acknowledgement, before deciding whether to retransmit the old block or pick a new one. This works fine, but it introduces pauses into the transmitted data stream that greatly reduce the data rate.

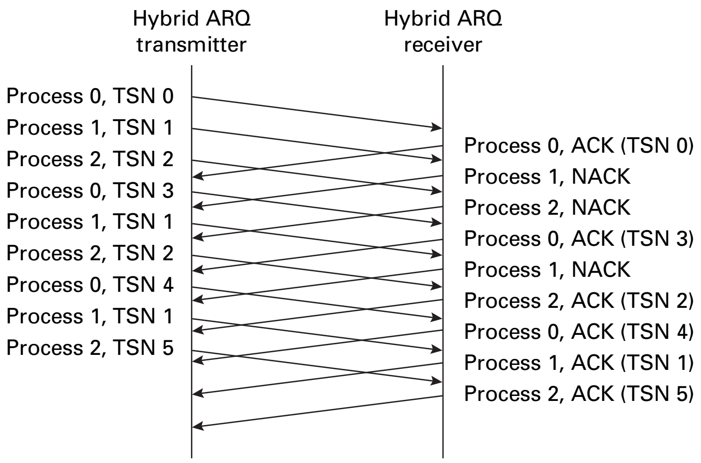

Figure 3.19 Operation of stop-and-wait retransmission when using three hybrid ARQ processes.

The solution is to use multiple parallel versions of Figure 3.18, each of which is known as a *hybrid ARQ process*. Every transmission time interval, the transmitter sends a block on one of the hybrid ARQ processes, while waiting for acknowledgements on the others. This allows the transmitter to send data in every transmission time interval, which maximises the data rate on the air interface. The effect, for the case of three processes, is shown in Figure 3.19. In the figure, positive acknowledgements are denoted ACK, while negative acknowledgements are denoted NACK.

The only problem is that different blocks may require different numbers of retransmissions, so they may emerge from the receiver's physical layer in a different order from the one expected. To handle this, the transmitter labels each block with a transmit sequence number, denoted TSN. In the receiver, the medium access control protocol inspects the transmit sequence number, and uses it to put the blocks back in the right order.

3.2.2 Fast scheduling

A second technique used for high speed packet access is fast scheduling of transmissions by the Node B. The technique is implemented a bit differently in HSUPA and HSDPA, so we will describe uplink fast scheduling here, and downlink fast scheduling later on as part of Section 3.2.4.

When a mobile is sending packet data, its data rate can be highly variable. The network therefore has to limit the rate at which each mobile transmits, to minimise the likelihood of a cell reaching its maximum data rate. In release 99, this is done by means of RRC signalling messages that are exchanged between the mobile and its serving RNC. Typically, the mobile indicates that it has data to transmit, and the SRNC responds by increasing its maximum permitted data rate. Unfortunately the signalling messages take a long time (typically hundreds of milliseconds), which makes the scheduling process inefficient. When a mobile reaches the end of its data stream, for example, there is a significant delay before its resources can be allocated to another one.

We can speed up the scheduling process by moving it to the Node B. The mobile sends the Node B a signalling message, to indicate the rate at which it would like to transmit. The Node B examines the mobile's request in the light of the current load in the cell, and reacts with a signalling message of its own, to indicate the maximum data rate at which the mobile will be allowed to transmit. The mobile transmits its data, and the Node B signals its acknowledgements as described above. This control loop does not involve the RNC, so it works much faster than the equivalent loop in release 99, and is much more responsive to changes in the network load.

If the cell is lightly loaded, then the scheduling process is easy: the Node B can just allow each mobile to transmit at the data rate that it requests. If the cell is congested, then the process is harder. At one extreme, the Node B can allocate the same maximum data rate to every mobile in the cell. At the other extreme, the Node B can maximise a cell's throughput by allocating a high data rate to the mobiles that can handle it, and a low data rate, or even zero, to the others. The usual strategy lies somewhere between these two extremes, so the scheduling algorithms are not defined by the standards; instead, they are proprietary to the equipment manufacturer or the network operator.

3.2.3 HSUPA

HSUPA was introduced in release 6. It combines the processes of fast scheduling and hybrid ARQ with soft combining, to increase the rate at which packet data can be transmitted on the uplink.

When a mobile is using HSUPA, it sends packet data on a new transport channel, called the *enhanced dedicated channel* (E-DCH). At the same time, the mobile continues to transmit on a release 99 dedicated channel (DCH), which handles power control and any circuit switched communications. The E-DCH uses a transmission time interval of either 10 ms or 2 ms: this is less than in release 99, which speeds up the scheduling process. It supports soft handover, but the E-DCH's active set can be a subset of the DCH's active set: a cell can be in the second of these but not in the first, to support cells that do not yet implement HSUPA. One cell in the E-DCH active set is nominated as the *serving cell*.

There are five new physical channels. The network uses the *E-DCH absolute grant channel* (E-AGCH) and the *E-DCH relative grant channel* (E-RGCH) to indicate the maximum data rate at which each mobile can transmit. More precisely, the serving cell indicates an absolute data rate on the E-AGCH, and can also adjust an earlier grant up or down using shorter messages on the E-RGCH. Non-serving cells only use the E-RGCH and only to reduce the rate at which a mobile can transmit. They do this when overloaded, so as to reduce the interference they receive.

The mobile transmits data on the *E-DCH dedicated physical data channel* (E-DPDCH). Depending on its capabilities, it can use up to two channelisation codes with a spreading factor of two, plus two channelisation codes with a spreading factor of four. It also sends physical layer signalling information on the *E-DCH dedicated physical control channel* (E-DPCCH). The best known signal is the *happy bit*, which is set to 1 if the mobile is happy with its current maximum data rate, or 0 if it would like to transmit at a higher data rate and has enough power to do so. The E-DPCCH also indicates the transmitted block size, and whether the block is a retransmission or a new one. Additional signalling is sent in the MAC header of the E-DPDCH, to give the Node B more information about the amount of data available for transmission.

Each Node B sends an acknowledgement on the *E-DCH hybrid ARQ indicator channel* (E-HICH). In the mobile, a hybrid ARQ process can move on to a new block if it receives a positive acknowledgement from just one Node B.

The medium access control protocol is more complex than in release 99. In the network, there are two new components. The MAC-e is implemented in the Node B. It schedules the mobiles' uplink transmissions, in response to their uplink signalling messages and the current load in the cell. The MAC-es is implemented in the serving RNC. It receives blocks of positively acknowledged data from the Node Bs, and puts them back in their original order. In the mobile, there is a single new component, known as the MAC-e/es.

3.2.4 Adaptive modulation and coding

The next technique we need to consider is *adaptive modulation and coding*. This is only used in HSDPA, and arises from a difference in the way transmit power is allocated in the uplink and downlink: in the uplink each mobile can work out its transmit power independently, but in the downlink the cell has to share its power among all the mobiles that it is transmitting to. That begs the following question: how does the cell allocate its transmit power in the best possible way? In particular, how should the cell respond if a mobile moves into a fade?

We have already seen the technique used in release 99: fast power control. If a mobile moves into a fade, then the Node B increases the power transmitted to it, so as to keep the received signal-to-interference ratio constant. This in turn allows the transmitted data rate to stay the same, in accordance with Equation (1.1). For real time, constant bit rate services such as voice, this is ideal. However, the extra transmit power is unavailable for the other mobiles in the cell, which might have used it more effectively.

For non-real time services, it is better to transmit to every mobile with a constant power, and to let the data rate vary. If a mobile moves into a fade (Figure 3.20), then the received signal-to-interference ratio drops, so the Node B has to reduce the data rate that it sends there. However, the

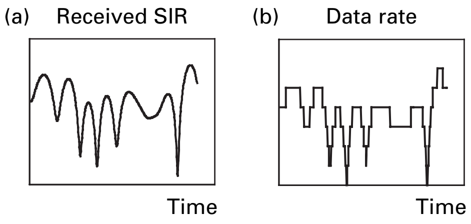

Figure 3.20 Operation of adaptive modulation and coding in HSDPA. (a) Received signal-to-interference ratio. (b) Transmitted data rate.

extra power is then available for the other mobiles in the cell. If the Node B uses that power for mobiles that are outside a fade, it can increase the data rate that it sends to them. Crucially, the cell's total data rate is higher than it was when using fast power control.

The process works as follows. The mobile measures the received signal-to-interference ratio, and indicates the maximum data rate it can handle using a physical layer signalling message on the uplink. The Node B then varies its transmitted data rate in two ways. In adaptive coding, it transmits with a constant spreading factor, but varies the amount of puncturing or repetition in the rate matching algorithm. By doing this, it varies the number of transport channel bits per frame, and hence the data rate. In adaptive modulation, it uses not only QPSK but also a modulation scheme known as 16-QAM (*quadrature amplitude modulation*). This uses a constellation containing 16 symbols instead of four, to transmit four bits in parallel instead of two. By switching between the two modulation schemes, the Node B is able to vary the transmitted data rate further.

The effect is a fast scheduling process, in which the Node B schedules its transmissions in response to signalling messages from the mobiles. As in the uplink, the scheduling algorithm is proprietary to the equipment manufacturer or the network operator.

3.2.5 HSDPA

HSDPA was introduced in release 5. It combines the techniques of hybrid ARQ with soft combining, fast scheduling, and adaptive modulation and coding, to increase the rate at which packet data can be transmitted on the downlink.

When using HSDPA, the mobile receives packet data on a new transport channel, which is known as the *high speed downlink shared channel* (HS-DSCH). As in HSUPA, the mobile continues to receive information such as circuit switched communications on a release 99 DCH. One difference from HSUPA is that the transmission time interval is always 2 ms. Another is that the HSDPA channels do not use soft handover: instead, all the communications are with the same serving cell that was singled out in the description of HSUPA. The main reason is that, on the downlink, the combination of soft handover with soft combining and fast scheduling would be extremely complex.

There are three new physical channels. The Node B sends data to the mobiles on the *high speed physical downlink shared channel* (HS-PDSCH). This uses a fixed spreading factor of 16, but it can be transmitted on up to 15 channelisation codes, which can be sent either to a single mobile or to many different ones. The Node B also sends the mobiles signalling messages on the *high speed shared control channel* (HS-SCCH). These alert mobiles to the imminent arrival of data on the HS-PDSCH; they also describe the characteristics of the data, such as the channelisation codes that will be used, the modulation scheme (QPSK or 16-QAM), and whether the data will be a retransmission or a new one.

A mobile sends acknowledgements to the network on the *high speed dedicated physical control channel* (HS-DPCCH). It also uses the HS-DPCCH to send signalling information known as *channel quality indicators* (CQIs): these indicate the highest data rate that the mobile can handle, given the signal-to-interference ratio that it is currently receiving. The Node B reacts to the channel quality indicator by adjusting its transmitted data rate on the HS-PDSCH.

There is one new component to the medium access control protocol, which is known as the MAC-hs. On the network side, this is implemented in the Node B, and schedules the Node B's transmissions in response to the mobiles' channel quality indications and the current load in the cell. In the mobile, the MAC-hs receives blocks of data from the physical layer, and re-orders them.

3.2.6 MIMO antennas

The final technique we will describe is *multiple input multiple output* (MIMO) antennas, which have been introduced in release 7 as part of HSPA+. In Section 1.3.4, we saw how a communication system could use multiple transmit or receive antennas for diversity processing, to increase the received signal strength and reduce the amount of fading. MIMO systems use multiple antennas at both the transmitter and the receiver, but for a completely different purpose: to increase the data rate. Figure 3.21 shows a MIMO system with two antennas at the transmitter and two at the receiver, which is the greatest number that the UMTS specifications currently support.

The transmitter divides the data stream in two, and sends half the data to each antenna. The receiver is configured as a *beamforming* system, which adds together the signals that arrive at the two receive antennas so that they interfere. Signals from some directions interfere constructively while others interfere destructively, so the effect is to synthesise a receive antenna beam that has maxima in some directions (where the interference is constructive) and nulls in others (where it is destructive). By

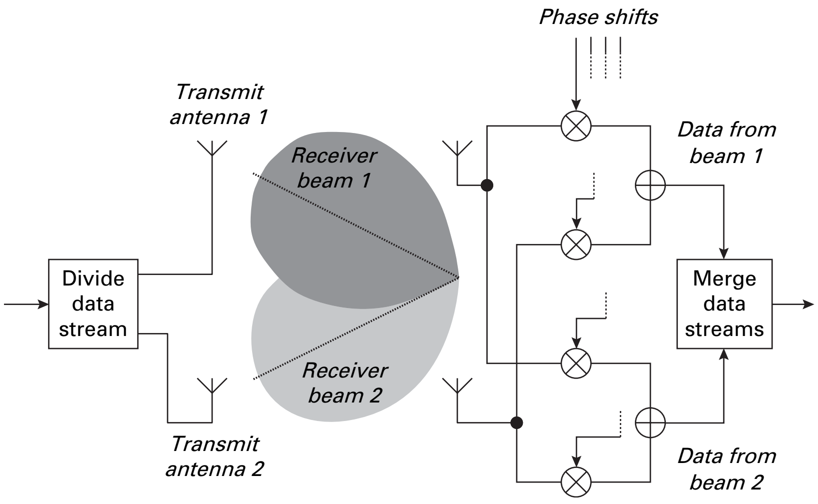

Figure 3.21 Simplified architecture of a MIMO antenna system.

applying additional phase shifts to the two signals before they are added, we can steer the nulls to any direction we choose.

In the MIMO receiver, we use two pairs of phase shifts to synthesise two independent antenna beams. We then steer beam 1 so that it has a null in the direction of transmit antenna 2, which ensures that it mainly receives data from transmit antenna 1. Similarly, we point a null of beam 2 towards transmit antenna 1, so that it mainly receives data from transmit antenna 2. We have then set up two nearly independent data streams, from antennas 1 and 2 in the transmitter to beams 1 and 2 in the receiver, so we can immediately double the data rate. This is, at least in theory, a highly effective way to get round the limit imposed by Equation (1.1).

It will come as no surprise to learn that this explanation is greatly simplified, and there are a few difficulties in the implementation of MIMO systems. The most important is fading. This introduces random, time dependent phase shifts into the rays that travel from transmitter to receiver, which makes it harder to work out the phase shifts to apply in the receiver. It turns out that we can only work out the phase shifts if the four fading patterns, from the two transmit antennas to the two receive antennas, are independent of each other. Any correlations between them degrade the performance of the MIMO receiver, and reduce the data rate that can be achieved.

3.2.7 HSPA+

Release 7 introduces a number of enhancements to high speed packet access, which are collectively known as *HSPA evolution* or *HSPA+*. The most important is the introduction of MIMO antennas for HSDPA, which implements the techniques described above with a maximum of two transmit and two receive antennas. HSPA+ also supports higher order modulation schemes. The uplink can transmit four bits in parallel using 16-QAM; the downlink can transmit six bits in parallel using 64-QAM, although it is limited to 16-QAM when using MIMO antennas.

Other enhancements reduce the mobile's power consumption at the times when it is not transmitting or receiving, and allow the HS-DSCH to be used without an accompanying DCH. Finally, there are improvements to the ways in which the network handles the data, which reduce the end-to-end delay and the delay jitter. This last set of improvements is particularly beneficial, as it reduces the delay jitter to values well below 100 ms, and makes high speed packet access usable for real time communications using voice over IP.

3.3 Performance of UMTS

The technologies described in Sections 3.1 and 3.2 allow UMTS to support higher data rates than earlier systems. Here, we will give an indication of the data rates that can be achieved. Two numbers are particularly important: the maximum data rate that a mobile can reach under ideal conditions, and the total data rate that a cell can typically handle. The first of these can be calculated directly from the specifications, and is the figure usually quoted in marketing literature. In practice, however, the second number is usually the one that limits the performance of the system, and is the main topic of this section.

We begin by discussing how the capacity of the uplink can be estimated in release 99. After describing the differences that appear in the downlink, we show how the capacity is increased by the use of high speed packet access, and summarise the advantages and disadvantages of CDMA compared with other multiple access techniques.

3.3.1 Behaviour of the CDMA uplink

The behaviour of the CDMA uplink is illustrated in Figure 3.22. The base station is trying to process the signal that it is receiving from the mobile on the left, against a background of interference received from all the other nearby mobiles. The interference arises because the other mobiles are using different scrambling codes from the first one, so their transmissions are uncorrelated but not orthogonal.

If there are only a few mobiles, then the interference levels are low, and the signal reception conditions are good. As the number of mobiles increases, so the interference levels increase. We can quantify the increase in interference using a number called the *noise rise*, NR, which

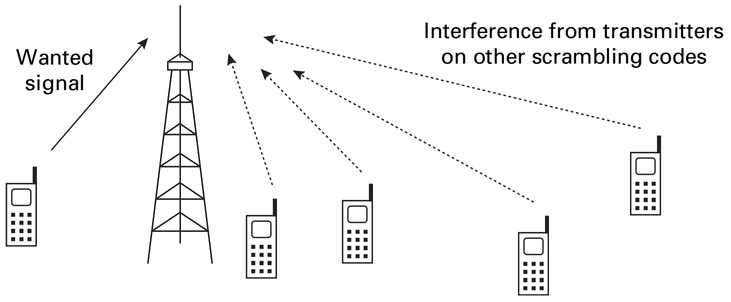

Figure 3.22 Origin of uplink interference in CDMA.

is defined as follows:

$$\mathrm{NR} = \frac{P_{\mathrm{total}}}{P_{\mathrm{thermal}}} \tag{3.2}$$

where P_{thermal} is the power at the base station receiver due to thermal noise, and P_{total} is the total power received from all sources, essentially thermal noise plus interference from all the UMTS mobiles.

In the uplink, the noise rise is relatively easy to estimate: some example calculations are in reference [1]. Figure 3.23a is a typical result, which shows how the noise rise varies with the total data rate in the cell, and hence with the total number of mobiles. As the total data rate increases, so the noise rise increases, first linearly and then increasingly quickly. Eventually, the noise rise becomes infinite at a data rate known as the *pole capacity*.

We can understand what is happening as follows. If the number of mobiles in the cell increases, so the interference levels at the Node B rise. The Node B can overcome the interference using the power control algorithm, by telling each mobile to increase its transmit power. Unfortunately this increases the interference further, and a vicious circle develops: each rise in interference requires more transmit power, which in turn leads to more interference. Eventually, at the pole capacity, the interference levels are so high that the base station can no longer hear any of the mobiles in the cell, no matter how powerfully they transmit. Thus the pole capacity is an absolute limit on the capacity of the cell.

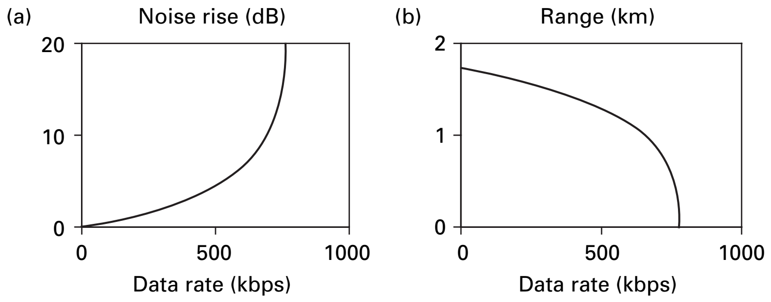

Figure 3.23 Behaviour of the UMTS uplink. (a) Example of how the noise rise varies with the cell throughput. (b) Example of how the uplink range varies with the cell throughput.

Interference affects the cell's range as well, as shown in Figure 3.23b. As the interference levels and the noise rise increase, so the allowed propagation loss falls, and the uplink range falls. By the time we reach the pole capacity, the range has dropped to zero: the base station cannot hear any of the mobiles, no matter how close they are. The variation of range with throughput in CDMA is often known as *cell breathing*. Cell breathing is not observed in systems that use FDMA or TDMA, where the cell's capacity is determined by the number of frequencies or timeslots, and is more-or-less independent of its range.

It is important to note that the pole capacity is not a fixed quantity. Instead, the uplink pole capacity can be estimated as follows:

$$T_{\max} \approx \frac{W}{(1+f)(E_b/N_0)} \tag{3.3}$$

where $T_{\max}$ is the uplink pole capacity and W is the chip rate. The number f depends on the cells' geometry: it is small if the cells are isolated, and large if they overlap and interfere with each other. E_b/N_0 is the minimum signal-to-noise ratio per bit required for satisfactory reception: it is low if the received signal power is constant and high if there is a lot of fading, and also varies a little with the target block error ratio and the underlying bit rate. In Figure 3.23, f is 0.6 and E_b/N_0 is 5 dB (i.e. a ratio of about 3.2), so the pole capacity in this example is about 760 kbps. In release 99, the pole capacity can vary between about 500 and 2000 kbps.

For a given pole capacity, the maximum number of mobiles in the cell depends on their average bit rate. If the pole capacity is 760 kbps, for example, then the cell can handle 62 mobiles at a rate of 12.2 kbps each, but only 11 mobiles at a rate of 64 kbps. This happens because a higher bit rate requires a higher transmit power to get the same energy into each bit, and this increases the interference that the base station receives. It is therefore better to think of the cell's capacity in terms of the total data rate it can handle, rather than the total number of mobiles in the cell.

The pole capacity is not a practical limit, because by the time the data rate reaches the pole capacity, the uplink range has dropped to zero. To deal with this problem, a cell normally operates using a *noise rise limit*. This is the maximum noise rise permitted within the cell, and leads to a practical capacity that is rather less than the pole capacity. If the noise rise starts to approach the limit, then the radio network controller has several ways of dealing with the situation. For example, it can reduce the bit rates of voice calls, delay the transmission of non-real time packet data, or block users when they try to make new calls.

In Section 3.1.3, we stated that the highest capability mobiles could transmit at a data rate of about 2048 kbps. This is usually greater than the capacity of the cell. A mobile can only reach such high data rates under ideal conditions: it must be the only mobile in the cell that is transmitting, it must be close to the base station to prevent it from hitting its maximum transmit power, and the pole capacity must be unusually high. These conditions can sometimes be met in picocells and even microcells, but are almost impossible to achieve in macrocells. As a result, it can be very difficult for a high capability mobile to reach its maximum data rate.

3.3.2 Behaviour of the CDMA downlink

The CDMA downlink is harder to model than the uplink, but the usual treatment runs as follows. We begin by assuming that the cell's range is limited by the uplink. This assumption is generally correct, because a base station can generally afford a powerful enough transmitter to handle all the mobiles in the cell. (In UMTS, most mobiles have a maximum power of 21 dBm ($\frac{1}{8}$ watt), while a macrocell's power is around 43 dBm (20 watts).)

We then estimate the downlink power required to reach each mobile, taking into account the interference from other base stations that are transmitting on different scrambling codes. By adding the individual power requirements, we can estimate the total power that the base station needs to transmit. If this is less than the power available at the base station, then the downlink will operate satisfactorily.

A typical result is shown in Figure 3.24, for conditions that are similar to the ones used earlier. As we saw on the uplink, the transmit power increases first linearly, and then increasingly quickly as interference comes to dominate. In this example, we reach the downlink pole capacity at a throughput of about 690 kbps, although we cannot quite get there because of the limit imposed by the base station's maximum power.

The downlink pole capacity is governed by a slightly different equation and, depending on the circumstances, can be greater or less than that of the uplink. However, the data rates on the downlink are usually greater than those on the uplink, because mobile data subscribers are likely to download far more data than they upload. As a result, we usually reach the downlink pole capacity more quickly. We can therefore summarise the behaviour of a UMTS cell as follows: coverage is usually limited by the uplink, but capacity is usually limited by the downlink.

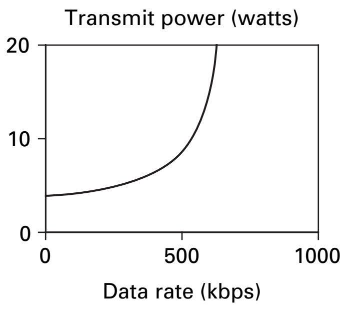

Figure 3.24 Example of how the transmitted power varies with the cell throughput on the UMTS downlink.

3.3.3 Performance of HSPA

High speed packet access increases both the maximum data rate of each mobile, and the pole capacity of each cell. However, the increase in the first is usually larger.

In HSUPA, a mobile has a maximum data rate of 5.7 Mbps. This is roughly three times greater than in release 99, but can only be achieved by configuring the rate matching algorithm to use a large amount of puncturing. In HSDPA, the maximum data rate is 14 Mbps. This is roughly six times greater than before, and is achieved by using heavy puncturing in conjunction with 16-QAM. In both cases, there are only a few extra bits available for error correction. This implies that a mobile can only reach these high data rates if it lies very close to the base station, so that the received signal power is very high.

High speed packet access increases the pole capacity as well. On both the uplink and the downlink, hybrid ARQ and soft combining allow the system to be operated at a higher block error ratio than in release 99. This reduces the transmitted signal power, so it reduces the amount of interference in the cell, and increases its pole capacity. On the downlink, adaptive modulation and coding allows the base station to direct its transmit power to the mobiles that can support a high data rate, and this increases the downlink capacity further.

The increase in pole capacity is hard to estimate, as it depends on the exact conditions being used. As an example, however, reference [2] suggests that HSUPA can increase a cell's capacity by about 30 to 90 per cent, and HSDPA can increase it by a factor of about 2 to 3. While significant, this is less than the increase in the peak data rate that we noted above. As a result, it is even harder for a high capability mobile to reach its maximum data rate when using high speed packet access than it was in release 99.

3.3.4 Advantages and disadvantages of W-CDMA

We close this chapter by summarising the main advantages and disadvantages of wideband CDMA, when compared with other multiple access techniques. There are three main advantages.

The first advantage comes from the fact that a cell's capacity is limited by interference. If a mobile is not transmitting, then it does not cause any interference to the base station, so it doesn't take up any uplink resources. Similarly it doesn't take up any downlink resources if it is not receiving. This is particularly useful in a voice call, where a typical mobile is only transmitting information for half the time, and receiving for the other half. The capacity improvement is not as great as it might be, because the mobile still has to transmit and receive continuously on the DPCCH. Despite this, the effect increases the cell's voice capacity by about 50 per cent.

Second, the short chip duration allows a receiver to do multipath diversity processing, using the rake receiver that we described earlier. This reduces the amount of fading, and allows the system to work with a lower received signal power than it otherwise would. In turn, this reduces the amount of interference, and increases the capacity of the system.

Third, the cells in a CDMA network are distinguished by the use of different scrambling codes, so they can all use the same carrier frequency. This is a different situation from the one in FDMA or TDMA, in which nearby cells have to use different carrier frequencies, and means that a CDMA network can accommodate more mobiles per unit bandwidth. Radio spectrum is a scarce and sometimes expensive resource, so this can be an important advantage.

CDMA does, however, bring a couple of problems. The first is related to radio network planning. Interference between neighbouring cells means that we cannot plan each cell independently, as we could in FDMA or TDMA. Instead, changes to one cell can affect the behaviour of cells nearby. If, for example, we increase the power transmitted from one cell in an attempt to increase its range, then that will increase the interference to the downlinks of neighbouring cells and degrade them. These inter-relationships make radio network planning a more complex process than it has previously been.

Another problem is related to intellectual property. CDMA technology is covered by a large number of patents, which require substantial royalties from product developers, and have been slowing the wider adoption of UMTS. Some of the licensing terms are the subject of ongoing legal action and, at the time of writing, the eventual outcome is still unclear.

References

1. H. Holma & A. Toskala, *WCDMA for UMTS: HSPA Evolution and LTE*, 4th edition (Wiley, 2007).
2. J. Sköld, M. Lundevall, S. Parkvall & M. Sundelin, Broadband data performance of third-generation mobile systems. *Ericsson Review*, **82**:1 (2005), 14–23.