3GPP/TDD Receiver Implementation using Matlab tool

1 Background

The goals of this exercise are:

- to understand the basic layer 1 3GPP/TDD transmission format
- to implement the receiver in MATLAB

1.1 Logical, Transport and Physical Channels

Before focusing on the physical layer, we briefly review the architecture of 3G systems starting from the network layer. This is important to understand some of the reasoning behind choices at the physical layer. Layers 1-3 for a futuristic IP-based system are shown in Figure 1. This is really the most complicated version of the system that one would find in a basestation (or *Radio Gateway* if we are talking about IP-based traffic). We assume that the network layer (i.e. IP) generates packets corresponding to different users or different streams for/from the same user, which may provide different services in parallel like voice and video, ftp and http, etc., which require different quality-of-service (QoS) measures. The IP layer can also generate signalling traffic which may be used to provide authentication, accounting or mobility management functionality. The IP network entities which control this type of traffic are collectively known as *AAA servers*. This control traffic will typically be mapped directly to an entity known as the *Radio Resource Controller (RRC)* which controls these functions directly using the specifics of the radio interface. In some sense, it provides the translation capacity from the IP world to the radio world. IP traffic from layer 3 is split (scheduled) into what are known as *radio bearers* based on routing and QoS constraints. Each radio bearer has an associated data rate and QoS parameters (e.g. packet loss rate, average delay, etc.) Some radio bearers will be suitable for real-time video, others for streaming audio/video or WWW browsing, and others for a variety of control/signalling functions and services.

The radio bearers make their way down to the physical layer via entities known as *Logical and Transport channels*. Logical and Transport channels can be simply thought of as pipes for different types of messages (e.g. broadcast, paging, dedicated user traffic, etc.) which are multiplexed and mapped onto physical layer resources using different scheduling and segmentation primitives. Logical channels are essentially queues which contain segmented and multiplexed network layer packets (i.e. IP packets). The logical channels pass through the radio link layer (RLC) which may append sequencing numbers to (sub)packets and apply a retransmission strategy (i.e. ARQ).

The logical channels are connected to *the medium access control (MAC) layer* which performs scheduling and rerouting of logical channels at its input. Its outputs are known as *transport channels*. A secondary function of the MAC layer is to choose the transport format of the transport channel, which essentially determines the amount of physical resources (bandwidth) that are to be allocated to the transport channel. This decision depends on the physical limitations and the amount of traffic that is required or generated by the channel. For instance, a user may request a transport channel which requires an average data rate of 200 kbits/s, but has strong variations in the required data rate at specific moments (i.e. WWW browsing). The MAC layer dynamically adjusts the radio resource allocation to provide a compromise between the user's requirements and the global system performance. The scheduling function at the MAC layer input and allocation strategy for transport formats are important determining factors of the radio-link QoS characterization since they impact the transmission delay statistics of the service.

In 3G systems[1] the primary transport channels and their functions are

- the *BCH or broadcast channel* : This is the most critical downlink transport control channel since it provides the heartbeat of the system. The basestation broadcasts basic cell information at regular intervals on this channel (i.e. frame structure, physical layer configuration data, etc.)
- the *FACH or forward-access channel* : This is the second most critical downlink transport control channel. It is used by the basestation to broadcast control information to users that are already attached to the system (i.e. their presence has been acknowledged by the system via the basestation)
- the *RACH or random-access channel*: This is an uplink transport control channel which is used by the mobile station to send short (or random) bursts of data to the basestation. It is a collision/contention channel which is mainly used during the initiation of a connection between the mobile terminal and the basestation. The mobile typically transmits at regular intervals its request for service on this channel and waits for a response from the basestation on the FACH.
- *the DCH or dedicated channel*: This is a uplink or downlink transport channel used for dedicated traffic data. A mobile terminal requests such a transport channel using a higher layer protocol sequence on the RACH/FACH.
- *the DCCH or dedicated control channel*: This is an uplink or downlink transport channel used in conjunction with a DCH to pass control information along with the data. Typically it could contain acknowledgment messages for a retransmission policy, or other such protocol signalling messages.
- *the PCH or paging channel*: This is a downlink transport channel which is used to send paging messages to mobile terminals. This is typically to awake the terminals when they are in low-power or idle modes.

Physical channels are the lowest level protocol entities used to convey user data or control traffic. They are mapped directly from raw bits originating in the transport channel blocks, to coded bits (e.g. from a convolutional code, CRC code and interleaving function), to modulation symbols and finally to the actual transmit signal itself. The front-end of the physical layer maps transport channels onto the different physical channels via an entity called a *composite coded transport channel (CCTrCH)*. From the point of view of the physical layer, this simply a multiplexing and coding of transport channel data which is segmented into chunks which can be mapped onto the physical channels. It contains a dynamic *rate-matching* function which adds (repetition) and removes (puncturing) coded bits to adjust the transport channel data size to physical layer constraints.

The Physical channels are indexed by their slot number orthogonal channelization (spreading) code. Spreading is used to allow several users or several streams for the same user (e.g. voice + video) to share a particular timeslot (i.e. CDMA multiple-access). At the same time it is also used as a means of achieving multi-level coding. On the downlink up to 16 parallel channelization codes can be used. Spreading is achieved through the use of Orthogonal Variable Spreading Factor (OVSF) codes which are based on Hadamard sequences. The allowed downlink spreading factors are 16 and 1, whereas on the uplink factors of 1,2,4,8,16 are permitted. The maximum number of users sharing a particular timeslot is 8, and users can use at most 2 channelization codes in parallel.

[1] Other system have very similar primitives for transport channels

1.2 3GPP/TDD Frame/Slot Structure

The frame structure of 3GPP/TDD implementation is shown in figure 2. We see that each frame is composed of 15 slots which can be dynamically allocated for either uplink or downlink transmission which allows for the possibility of asymmetric resource allocation for the two traffic directions. The basic transmission entity in a slot is a burst. A burst is comprised of

- physical channels
  - dedicated data
  - control data
  - primary/secondary synchronization
  - basestation synch
- a midamble
- a guard-interval
- Transport Format control bits (TFCI)
- Power Control bits (TPC)

A typical burst (corresponding to the PCCPCH+SCH, see below) is also shown in Figure 2. Bursts are further subdivided into 2560 QPSK symbols (chips). The four basic burst types are shown in Figure 8. Burst types 1 and 2 are for standard data and control channels. The long midamble (512 chips) is used when up to 8 uplink users share the channel, so that the basestation can jointly estimate the 8 users' impulse responses. This burst type can potentially also be used with up to 8 downlink transmit antennas, to allow the users to estimate the channel impulse responses received from each of the 8 antennas. The length 256 midamble in Burst type 2 allows for up to 3 parallel channels to be estimated jointly.

The midambles are the superposition of either 3 or 8 cyclic shifts of a training sequence with a periodic extension. This structure allows for efficient channel estimation techniques based on the *Fast Fourier Transform (FFT)*, as is explained in Appendix 5.2. The length 256 midamble is shown in figure 4, where each letter in the figure corresponds to a sequence of length 64 chips. We see that the last 64 chips of each shift is the same as the first 64 chips. This is the periodic extension mentioned earlier and explained in Appedix 5.2 (in the appendix it corresponds to the N_e symbols added to the training sequence). The basic perdiod of the midamble is 192 chips and corresponds to L in Appendix 5.2.

The PRACH (Physical Random Access Channel, see below) burst is similar to burst type 1, except that it has a longer guard period. This is because the channel is used for initial communication from the mobile terminal to the basestation where timing adjustments due to propagation delay have not yet been made. The extra guard symbols account for a possible long delay. Based on the delay computed by the basestation, the mobile terminal is instructed to transmit early when it uses normal channels (i.e. Burst types 1 and 2).

The Node-B synchronization burst is used to synchronize basestations (or Node-B's) of adjacent cells in the network. Basically, one Node-B is configured to act as a master from time to time and transmits in the timeslot allocated to the PRACH (where it normally supposed to be listening). The Node-B's from surrounding cells are listening and can synchronize themselves to the (temporary) master. This allows for all adjacent cells to be frame synchronous without having to distribute clocks in the network or rely on common synchronization signals such as those that can be derived from the GPS (global positioning system).

1.2.1 Channelization Codes

Channelization (spreading) codes are simply an organization of Hadamard codes of varying lengths which allow users (or data streams of one user) of potentially different data rates to be mutually orthogonal. These are known as OVSF codes. Figure 5 shows the OVSF code tree for spreading factors 1,2, and 4. Each level of the tree contains the rows of a Hadamard matrix. In addition to being orthogonal to all sequence at the same level, the organization of the tree allows any sequence to be orthogonal to any other sequence which is not a direct ancestor or descendant in the tree (this is easily verified.)

1.2.2 Scrambling codes

Scrambling is used to reduce interference from signals originating in neighbouring cells. Each cell is assigned a scrambling sequence of length 16 chips. This is a major difference from other CDMA systems such as UMTS/FDD and IS-95 which use very long scrambling codes. Because of this short scrambling sequence, advanced receiver architectures (i.e. equalizers, multiuser receivers) can be implemented at reasonable computational cost.

1.2.3 Modulation Pulse-Shaping

QPSK signaling is used in 3GPP/TDD. Pulse-shaping is achieved through the use of a root-raised cosine filter with a spectral roll-off factor of .22. The resulting channel bandwidth is 5 MHz.

1.2.4 Dedicated Physical Channels (DPCH)

Dedicated physical channels are used to transmit user traffic. These are typically a multiplexing of DCH and DCCH transport channels. Any combination of burst types, channelization codes and TFCI formats are allowed. On the downlink the allowed spreading factors are 16 and 1, whereas on the uplink factors of 1,2,4,8,16 are possible.

1.2.5 Primary common control physical channel (PCCPCH)

The *BCH* or *broadcast channel* is mapped onto the *Primary Common Control Physical Channel (PCCPCH)*. Because of its critical nature in the system (i.e. the mobile has to at least be able to demodulate this channel) it is given special attention. The PCCPCH occupies one timeslot per frame and is the only channel to occupy the timeslot. The position (time slot / code) of the P-CCPCH is known from the Physical Synchronisation Channel (SCH). This channel always uses a spreading factor of 16 and channelization code 0. Burst type 1 is used without TFCI. It is the only physical traffic channel for which its physical configuration is uniquely determined.

1.2.6 Secondary common control physical channel (SCCPCH)

The PCH (Paging Channel) and FACH (Forward Access Channel) are mapped onto one or more *Secondary Common Control Physical Channels (SCCPCH)*. Each SCCPCH can use either a spreading code of 16 only. Since more than one can be used in parallel, the capacity of PCH and FACH can be adapted to different system requirements. In order to demodulate the SCCPCH, the mobile must first determine the physical layer characteristics by demodulating the system information broadcast on the BCH since the spreading codes are cell dependent and can change on a regular basis.

1.2.7 The physical random access channel (PRACH)

The RACH (Random-access Channel) is mapped onto one uplink physical random access channel (PRACH). This channel is used by the mobile station during the connection phase with the basestation and for passing control information once connected. The spreading factor of this channel can be either 8 or 16 and the allowable spreading codes are broadcast on the system information. This is a collision/contention channel and therefore the spreading code and transmission time are chosen at random by the higher layers in order to reduce contention. Up to 8 users can be supported by the basestation at any given time.

1.2.8 Primary and Secondary Synchronization (SCH)

The synchronization channel, SCH, is a physical channel which is not related to any particular transport channel. It is used for two purposes in UMTS/TDD. First to acquire initial frame synchronization (using the primary synchronization code) and second to derive the code group of a cell (using the secondary synchronization code). In order not to limit the uplink/downlink asymmetry the SCH is mapped on one or two downlink slots per frame only.

There are two cases of SCH and PCCPCH allocation:

1. SCH and P-CCPCH allocated in timeslot $k, k = 0..14$
2. SCH allocated in two timeslots: timeslot k and $k + 8, k = 0.6$; P-CCPCH allocated in timeslot k.

The position of SCH (value of k) in frame can change on a long term basis in both cases. In either case, detection of the SCH yields the position of the PCCPCH within the frame. Figure 6 is an example for transmission of SCH, $k = 0$, of Case 1.

As depicted in 6, the SCH consists of a superposition of a primary, $s_p[k]$, and three secondary code sequences, $s_i[k], i = 0, 1, 2$ each 256 chips long. Typically a mobile station will detect several candidate base stations using the primary synchronization sequence and select the one with the strongest signal. Since base stations are required to be synchronized, the time-offset of the SCH is used to ensure that the SCH from neighbouring cells do not overlap in time, thus allowing the mobile station to distinguish them. The time offset t_{offset} is one of 32 values, depending on the *code group* of the cell. The secondary sequences carry 6-bits of information since they are each modulated by a QPSK symbol, $b_i, i = 0, 1, 2$. The information conveyed by the secondary sequences is used by the mobile terminal for determining the code group (and thus t_{offset}.) The code group contains information about the scrambling code and midambles to be used in the cell covered by the basestation. The remaining bit of information allows the mobile terminal to determine the frame number modulo-2. This is needed to correctly pass the demodulated PCCPCH data to the channel decoding functions and the higher layers, since the BCH data found on the PCCPCH is coded and interleaved across 2 radio frames.

2 Basic Transmitter structure

The implementation of the transmitter used here can generate a composite signal containing up to 16 variable-rate, variable power data streams per slot. It is shown in figure 7. The rates of the different streams is controlled by OVSF spreading sequences $\phi_i[n]$ is the and the amplitudes by A_i. It does not include the complex scrambling described in the 3GPP specifications. The choice of $\phi_i[n]$ dictates the spreading factor L_i, which ranges from $2^n, n = 1..4$.

The composite signal is upsampled by factor 2 and filtered by a 12-tap root-raised cosine FIR filter, $p[k]$. The end result is a complex baseband 3GPP TDD signal at two samples per chip.

3 Constructing the Receiver

In the MATLAB exercise we will construct the receiver to do the following

- perform frame synchronization using the primary synchronization signal of the SCH
- estimate the impulse response of the channel using the FFT method on the midamble
- synthesize a matched filter and demodulate the data portion of the PCCPCH

The basic receiver structure for the mobile terminal is shown in Figure 8.

3.1 Frame Synchronization

Frame sychronization is achieved using a filter matched to the primary synchronization sequence to estimate the location of the start of a frame. The generic technique is described in Appendix 5.1. It is achieved by filtering the bandpass received signal as

$$r_s[n] = \left\{ \sum_{k=0}^{255} s_p[k]\delta(n-2k) \right\} * r[n] = \sum_{k=0}^{255} s_p[k] r(n+2k) \tag{1}$$

and averaging the $r_s[n]^2$ over several frames. In the real-time receiver, the maximum output of this filter is used to adjust the receive signal strength (via a variable gain IF amplifier) and synchronization is achieved when the maximum is greater than a pre-defined threshold. Although no multiplications are required, this is typically the most computationally intensive part of the receiver front-end since it requires a fairly long filter operating at the sampling rate. The 3GPP standard uses a hierarchical structure which allows the filter to be implemented as a concatenations of 2 FIR filters of length 16, which use only additions and subtractions (i.e. the filter coefficients are ± 1).

3.2 Channel estimation

Once frame synchronization is achieved, the location of the midamble can be determined (from the Type 1 burst format) and the impulse response from the basestation to the mobile can be estimated. This operation is performed once per slot. Channel estimation is performed as described in Appendix 5.2 using the type 2 midamble (256 chips).

3.3 Matched Filtering

Matched filtering is done by first synthesizing the matched filter by convolving the complex channel estimate and the chosen spreading sequence, $\phi_0[k]$. For the PCCPCH, the spreading sequence has index zero and spreading factor 16. In the real system, the spreading sequence is also scrambled using the scrambling code from the cell. Once the matched filter coefficients are determined, the projection of the matched filter on the received sequence is performed for every data symbol in the slot.

4 Work to be done

You will be asked to form groups of no more than 3 students. Everyone will implement the entire receiver for slot 0 (PCCPCH) on the signals found in the work directory. You will

1. perform the primary synchronization detection to locate the PCCPCH (assume code group 0 and $t_{\text{offset}} = 0$). We have provided the primary synchronization sequence.

2. estimate the channel for the PCCPCH. We have not implemented the channel as specified by 3GPP since we use a midamble of length 256. The chosen midamble is provided to you. Provide plots of the various channel estimates.

3. generate the matched filter for $\phi_0(k)$ (spreading 16, OVSF index 0) (you may make use of the supplied MATLAB function `ovsfgen.m`)

4. demodulate the BCH data on the PCCPCH using the matched filter. Provide plots of the matched filter output (you may use the MATLAB function `stem` on the real-part of the signal, and plot the complex scatter diagram output using `plot(x,y,'.')`).

You will hand in the following:

1. signal plots of all intermediate signals (i.e. synch output, channel estimates, spectrum)

2. signal plots of the matched filter outputs

3. your MATLAB code

4.1 Supplied MATLAB functions

One MATLAB function is supplied for generating the OVSF spreading codes. It is called as `ovsfgen(SF,index)` where `SF` is the spreading factor and `index` is the index into the OVSF tree for the chosen spreading factor.

5 Appendix

5.1 Frame Synchronization for Spread-Spectrum Signals

In this section we assume that each transceiver keeps track of the passage of time by means of identical copies of a clock. All clocks have the exact same frequency, but they are started at different times. This means that everyone agrees on when a certain amount of time has passed but they don't agree on the absolute time.

The time difference is a problem since the idea of TDD and TDM relies on placing the information in slots and agreeing on which slots to use for what. But the identity of a slot is specified by its position in time. We need to make sure that the frame boundaries of a user line up with those of all the other users within a 5 MHz band.

Things are slightly simpler if there is a master. So let us assume that there is a master (the base station in cellular communication). We are done if we ensure that all the slaves (the mobile stations in cellular communications) line up their frame to that of the master. They all know how long a frame is but they don't know where one (and thus all) frame begins due to time offsets.

For the purpose of this discussion we can assume that there is no propagation delay. We'll come back to it later.

For the purpose of this discussion we can assume that there is no propagation delay. We'll come back to it later.

To make it easier for the slaves, the master sends a synchronization signal. The signal is repeated every frame at a fixed location. We may assume that the synchronization signal starts at the beginning of the first slot of each frame. If a slave can detect the beginning of the synchronization signal, then it also knows the beginning of the frame. When this is the case we say that the slave has achieved frame synchronization. Let the synchronization signal be

$$s(t) = \sum_i s_0(t - iT_f)$$

where T_f is the duration of a frame.

The receiver (a slave) observes

$$r(t) = s(t - \tau) + n(t)$$

where, with a slight about the notation, t represents the local time of the slave, τ is the unknown time offset with respect to the master, and $n(t)$ is additive white Gaussian noise with power spectral density $N_0/2$.

The slave wants to learn τ or, equivalently, the beginning of the waveform s_0 within the received signal.

Let $\rho_0(\xi)$ be the auto-correlation of s_0. Recall that an auto-correlation function has its maximum at 0 and the value of this maximum is the signal's energy. Ideally, we would like to use a signal s_0 for which $\rho_0(\xi)$ vanishes everywhere except at 0. White noise has this property. Let us assume for now that the auto-correlation of s_0 can be approximated by that of white noise, i.e.,

$$\rho_0(\xi) = \delta(\xi).$$

The slave correlates the received signal with $s_0(t)$. This yields

$$\begin{aligned}
\rho(\xi) &= \int r(\alpha + \xi) s_0(\alpha) d\alpha && (2)\\
&= \int \left[\sum_i s_0(\alpha - iT_f - \tau + \xi) + n(\alpha + \xi) \right] s_0(\alpha) d\alpha && (3)\\
&= \sum_i \int s_0(\alpha - iT_f - \tau + \xi) s_0(\alpha) d\alpha + \int n(\alpha + \xi) s_0(\alpha) d\alpha && (4)\\
&= \sum_i \rho_0(\xi - \tau - iT_f) + n_0(\xi) && (5)\\
&= \sum_i \delta(\xi - \tau - iT_f) + n_0(\xi), && (6)
\end{aligned}$$

where $n_0(\xi)$ is the projection of white Gaussian noise onto $s_0(t + \xi)$. This is Gaussian noise.

The good news is that $\rho(\xi)$ has a delta function at values of ξ corresponding to $\ldots, \tau, \tau + T_f, \tau + 2T_f \ldots$. The position of a delta, and thus the value of τ, are easy to detect, even in the presence of additive noise.

The procedure that we have just described may seem ad-hoc. One may proceed more formally and set up the problem as a standard estimation problem where one has to estimate τ from observing $r(t)$. If we assume that τ is uniformly distributed in some interval of duration T_f, then the solution that we have described is optimal in the sense of minimizing the mean squared error $E[\tau - \hat{\tau}]^2$.

So far we have assumed that the baseband equivalent channel has impulse response $h(t) = \delta(t)$. More realistically, we should model the received signal as

$$r(t) = As(t - \tau) + n(t)$$

where A is some unknown (complex-valued) constant. Then

$$\rho(\xi) = \sum_i A\delta(\xi - \tau - iT_f) + n_0(\xi).$$

It is still easy to find the position of the delta Diracs by looking at where $|\rho(\xi)|$ achieves its highest values.

Often the signal propagates from the transmit to the receive antenna along many paths, each path having its own propagation delay τ and attenuation A. By the superposition principle, the jth path will contribute to $\rho(\xi)$ with a term of the form $\sum_i A_j\delta(\xi - \tau - iT_f - \tau_j)$. (Without loss of generality we may assume $\tau_1 = 0$). There are now many peaks of $\rho(\xi)$, each delayed and scaled according to τ_j and $|A_j|$, respectively. Typically we will choose the position of the strongest peak as the beginning of the frame.

Once it has achieved frame synchronization, a slave may start sending, provided it knows which slot to use. However, to receive it needs to estimate the channel, which is the subject of next section.

5.2 Channel Estimation

Recall that the ML receiver for the AWGN channel takes the inner product of the received signal with each of the possible transmit signal. If the channel has an impulse response $h(t)$, then the only change is that we need to use the signals as they appear after they go through a noise-free channel with impulse response $h(t)$. The point is that we need (an estimate of) $h(t)$. Clever implementations of a ML receiver do not need to perform all the inner products, but they still need to know (an estimate of) $h(t)$. Finding such an estimate is the *channel estimation* problem.

The channel estimation problem is similar to frame synchronization, except that it is usually performed once frame synchronization is achieved. Again we have

$$r(t) = s_0(t) * h(t) + n(t),$$

with the goal now being to accurately estimate the impulse response $h(t)$.

It is simplest to use a frequency-domain representation of $r(t)$, so we may write

$$R(f) = S_0(f)H(f) + N(f)$$

If the noise is white and Gaussian, each spectral component of $R(f)$ is independent of every other spectral component. This is the motivation for using a frequency-domain approach, since we can consider each spectral component separately.

For each spectral component we have a linear estimation problem and the optimal estimate (maximum-likelihood) of $H(f)$ is

$$\hat{H}(f) = \frac{R(f)}{S_0(f)}$$

In the time-domain we have $\hat{h}(t) = \int_{-\infty}^{\infty} \hat{H}(f)e^{-j2\pi ft}dt$. We see that using the frequency-domain representation yields a very simple and intuitive method for estimating the channel impulse response. We now discuss an efficient discrete-time implementation based on the FFT.

5.2.1 An FFT-based implementation

Suppose now that the known signal $s_0(t)$ is bandlimited. We may, therefore, consider the samples

$$r(k) = s_0(k) * h_{\mathrm{LP}}(k) + n(k)$$

where $h_{\mathrm{LP}}(k)$ is the sampled lowpass-filtered version of $h(t)$ (i.e. filtered in the bandwidth of $s_0(t)$.) Now consider the case where a window of length N samples of $s_0(k)$ is chosen, say $k = 0, ..., N-1$, and that $s_0(k) = s_0(k \mod L)$, where $L = N - N_e$ is some period and N_e is a *periodic extension*. This just means that the last N_e samples of $s_0(n)$ are the same as the first N_e samples. For this special case, we have that for any window of L samples

$$s_0(k) * h_{\mathrm{LP}}(k) = s_0(k) \overset{L}{\otimes} h_{\mathrm{LP}}(k), k \in [m, m+L-1]$$

where $\overset{L}{\otimes}$ denotes *circular convolution with period* L[2] We now recall the property of the DFT (Discrete Fourier Transform)

$$\mathrm{DFT}(x(k) \overset{L}{\otimes} y(k)) = \mathrm{DFT}(x(k)) \cdot \mathrm{DFT}(y(k)) = X(k)Y(k)$$

We see that the use of a periodic extension allows us to perform a DFT of length L samples on the received signal. This is important since for $N = \prod_l p_l^{M_l}$, where p_l is a sequence of prime numbers and M_l a sequence of powers, there exist efficient computational algorithms known as collectively as the FFT (*Fast Fourier Transform*) to compute the discrete-time channel estimate $\hat{h}(k)$. We simply perform

$$\hat{h}(k) = \mathrm{DFT}^{-1}([\mathrm{DFT}(s_0(k))]^{-1} \cdot \mathrm{DFT}(r(k)))$$

5.2.2 An extension for Multiuser Signals

Suppose now we consider the joint estimation of many channel impulse reponses in a composite signal. This is the case, for instance, in the receiver of a basestation receiving the superposition of many signals from mobile terminals. In general, we have

$$r(k) = \sum_{i=0}^{U-1} s_i(k) \overset{L}{\otimes} h_i(k) + n(k)$$

where U is the number of users, $s_i(k)$ is a known sequence for user i and $h_i(k)$ is the impulse response for user i. Let us now impose the following constraint on the $s_i(k)$:

$$s_i(k) = s_0((k - iN_e) \mod L)$$

Here we say that each of the known signals are cyclic shifts of the first sequence $s_0(k)$. We may now write

$$r(k) = \sum_{i=0}^{U-1} s_0((k - iN_e) \mod L) \overset{L}{\otimes} h_i(k) + n(k) = \sum_{i=0}^{U-1} s_0(k) \overset{L}{\otimes} h_i((k - iN_e) \mod L) + n(k)$$

[2] recall that $x(k) \overset{L}{\otimes} y(k) = \sum_l x(l)y((k-l) \mod L)$.

and use the previous technique to estimate

$$\hat{h}(k) = \sum_{i=0}^{U-1} \hat{h}_i((k - iN_e) \mod L).$$

This estimate is useful if we further assume that the duration of each impulse response is less than N_e so that impulse responses do not overlap[3]. The estimate of user i's impulse response is, therefore,

$$\hat{h}_i(k) = \begin{cases} \hat{h}(k + iN_e), & k = 0, 1, \cdots, N_e - 1 \\ 0, & \text{otherwise} \end{cases}$$

The length of the periodic extension, N_e, is chosen in practice to be at least as long as the worst-case time-delay spread of the typical propagation environments of the system.

[3]note that the duration cannot be strictly time-limited, since the signal is assumed to be bandlimited. This means that the overlap between the individual impulse responses will introduce a minimal amount of distortion in the estimates

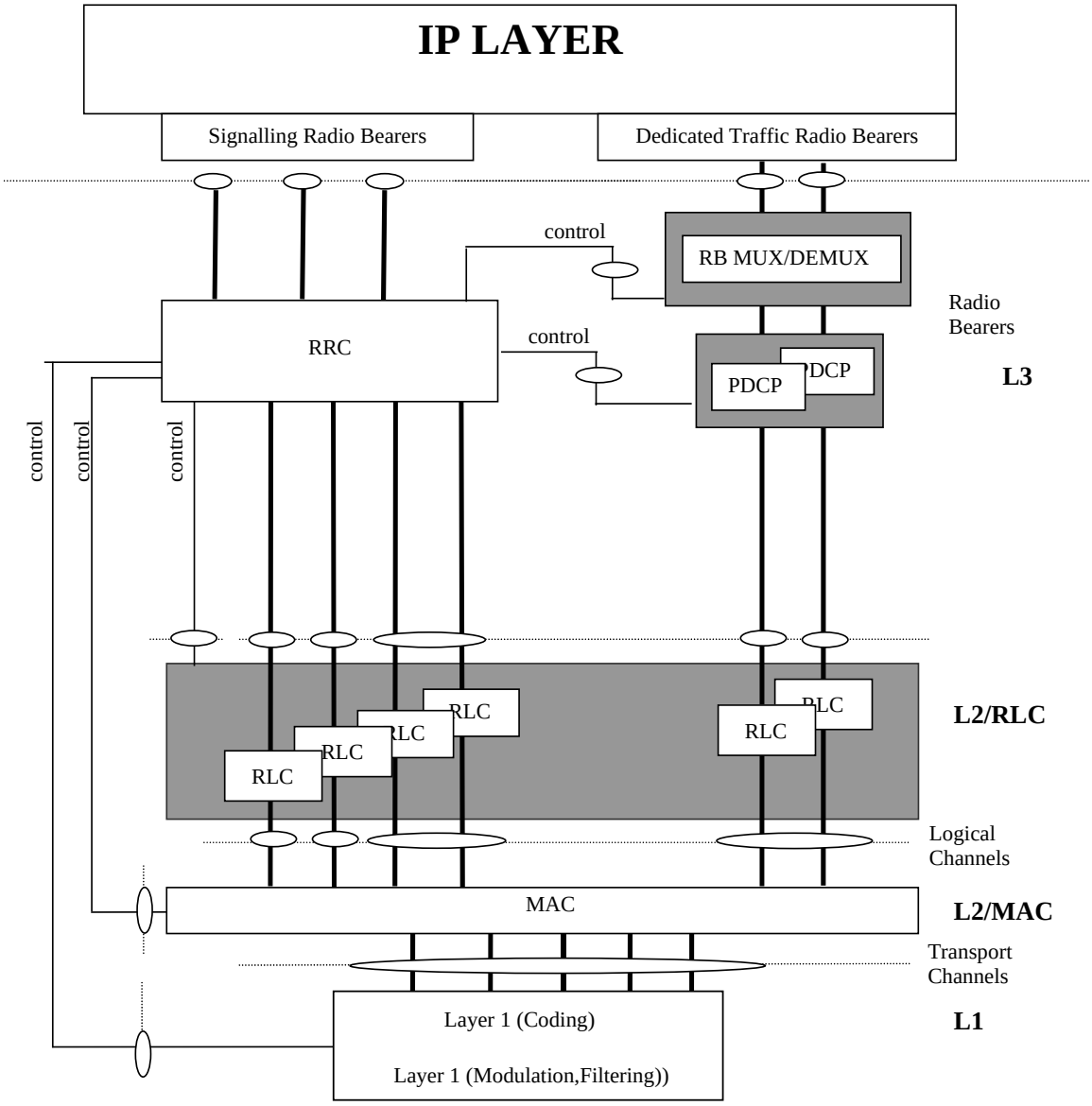

Figure 1: Radio Protocol Layers

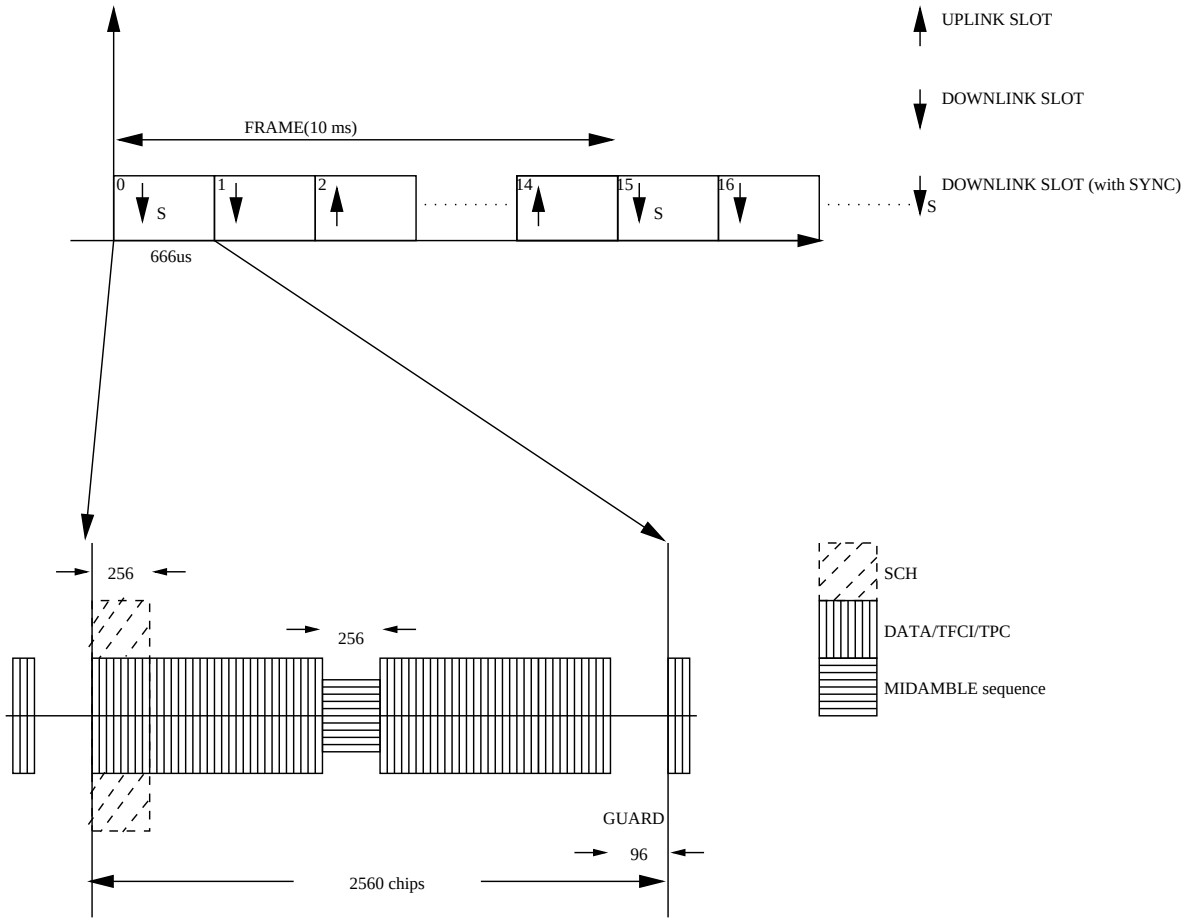

Figure 2: Frame structure

BURST TYPE 1

DATA A	TFCI A	MIDAMBLE (512 Chips)	TFCI B	TPC	DATA B	GUARD (96 chips)

BURST TYPE 2

DATA A	TFCI A	MIDAMBLE (256 Chips)	TFCI B	TPC	DATA B	GUARD (96 chips)

PRACH BURST

DATA A	TFCI A	MIDAMBLE (512 Chips)	TFCI B	TPC	DATA B	GUARD (192 chips)

NODE-B SYNCH BURST

CELL SYNC (2304 Chips)	GUARD (256 chips)

Figure 3: 3GPP TDD 3.84 Mchip/s Burst Structures

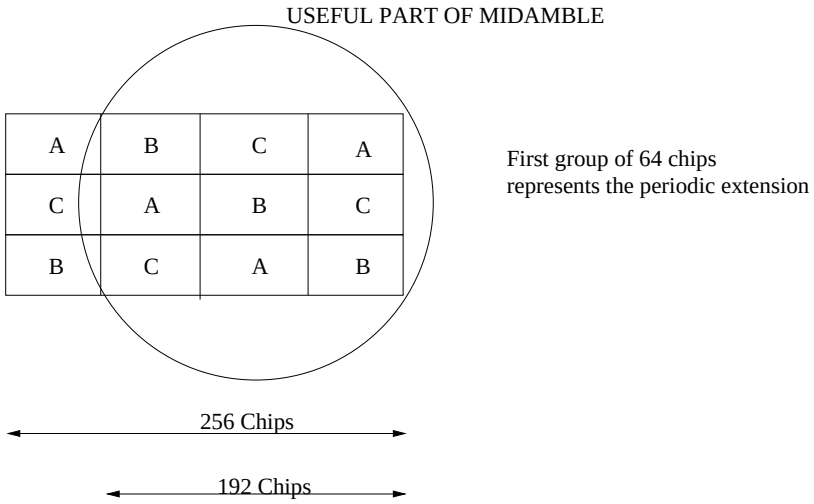

Figure 4: Midamble Structure (256 Chips)

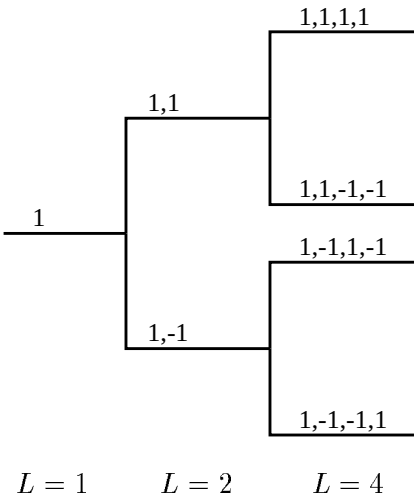

Figure 5: OVSF Code Tree

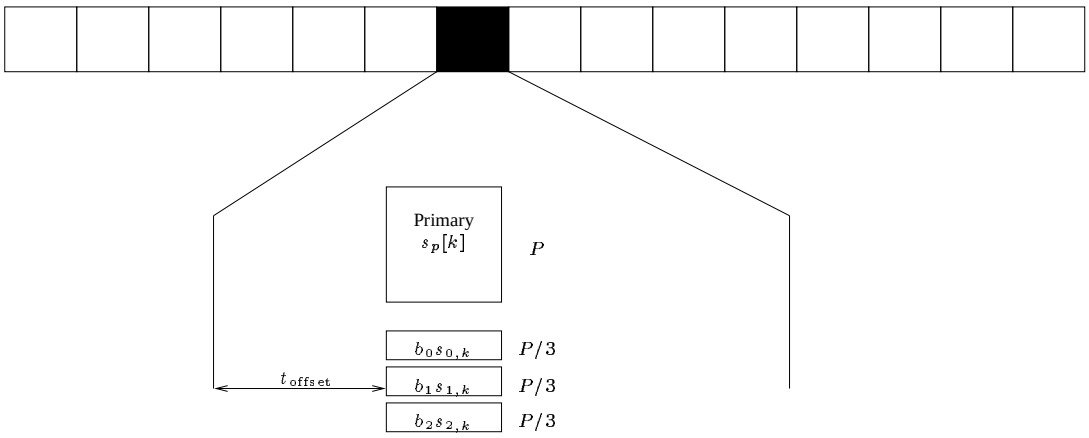

Figure 6: SCH signaling

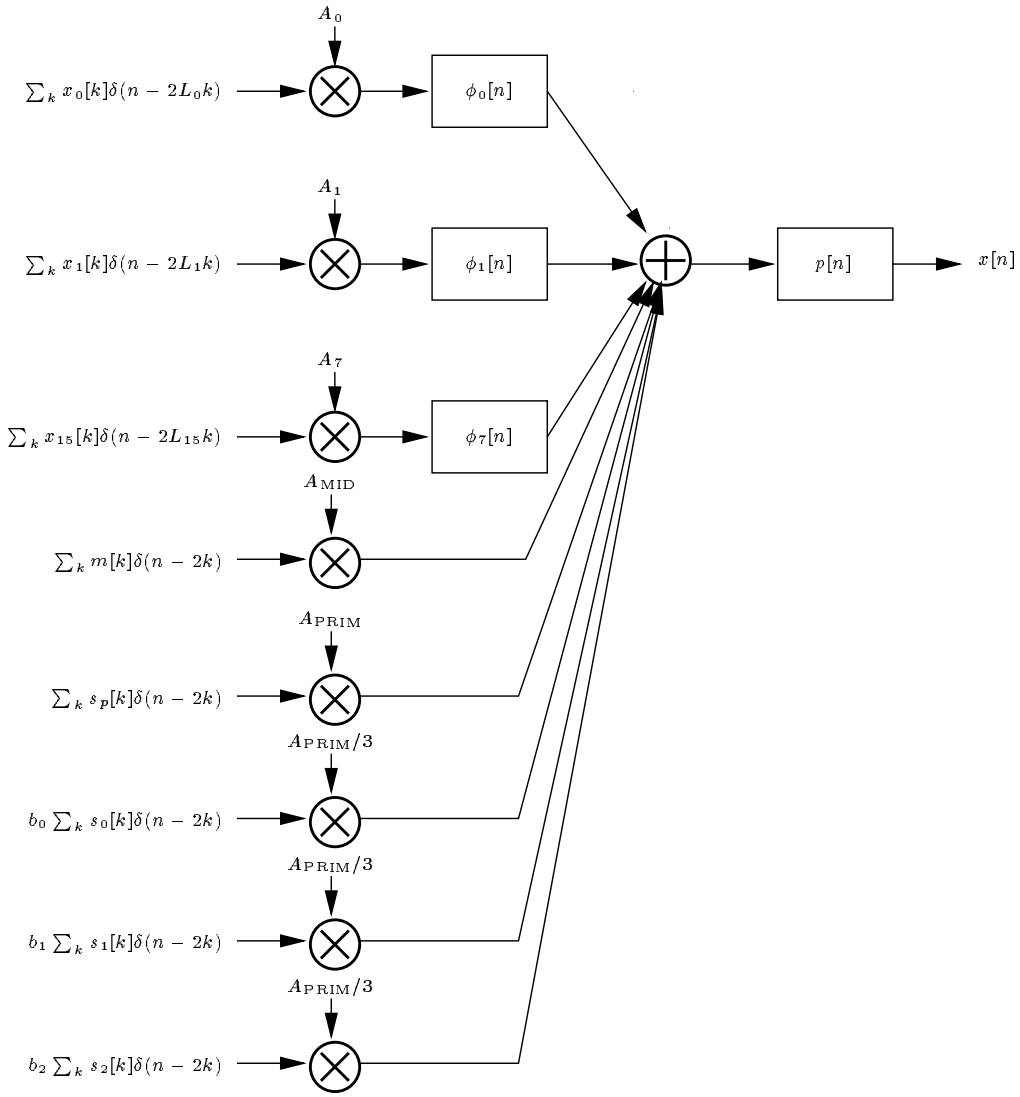
A_0
$\sum_k x_0[k]\delta(n - 2L_0k)$
$\phi_0[n]$
A_1
$\sum_k x_1[k]\delta(n - 2L_1k)$
$\phi_1[n]$
$p[n]$
$x[n]$
A_7
$\sum_k x_{15}[k]\delta(n - 2L_{15}k)$
$\phi_7[n]$
A_{MID}
$\sum_k m[k]\delta(n - 2k)$
A_{PRIM}
$\sum_k s_p[k]\delta(n - 2k)$
$A_{\mathrm{PRIM}}/3$
$b_0 \sum_k s_0[k]\delta(n - 2k)$
$A_{\mathrm{PRIM}}/3$
$b_1 \sum_k s_1[k]\delta(n - 2k)$
$A_{\mathrm{PRIM}}/3$
$b_2 \sum_k s_2[k]\delta(n - 2k)$

Figure 7:

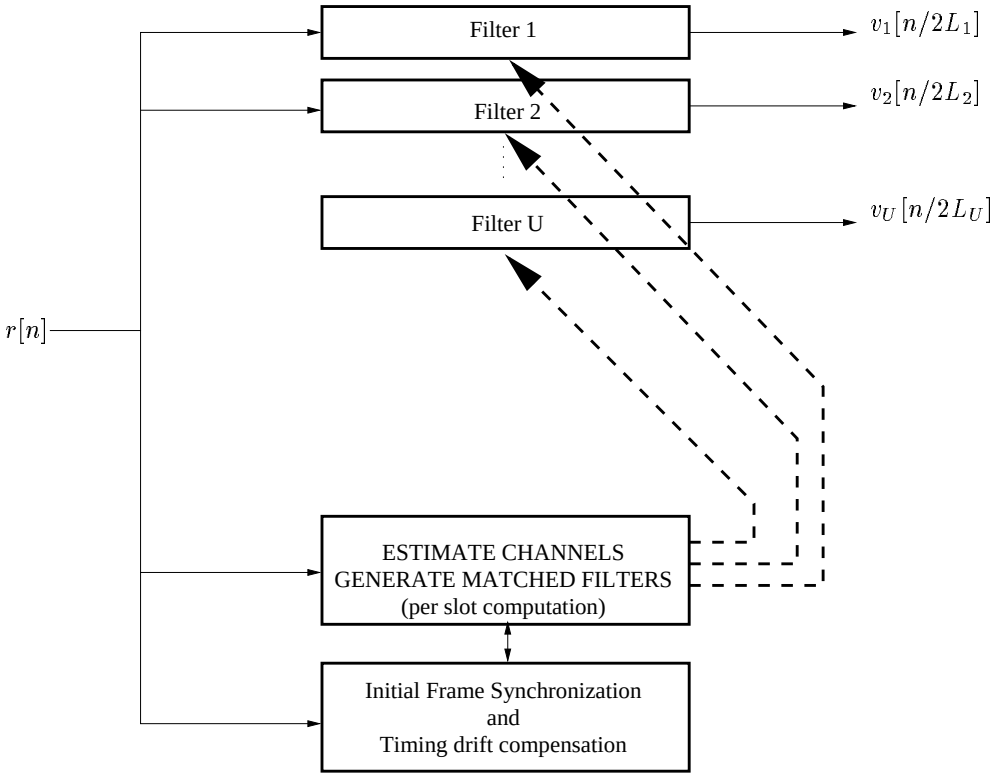

Figure 8: Receiver front-end