1 Introduction and objectives

This book focuses on the system-level engineering aspects of digital point-to-point communication. In a way, digital point-to-point communication is the building block we use to construct complex communication systems including the Internet, cellular networks, satellite communication systems, etc. The purpose of this chapter is to provide contextual information. Specifically, we do the following.

(i) Place digital point-to-point communication into the bigger picture. We do so in Section 1.1 where we discuss the Open System Interconnection (OSI) layering model.
(ii) Provide some historical context (Section 1.2).
(iii) Give a preview for the rest of the book (Section 1.3).
(iv) Clarify the difference between analog and digital communication (Section 1.4).
(v) Justify some of the choices we make about notation (Section 1.5).
(vi) Mention a few amusing and instructive anecdotes related to the history of communication (Section 1.6).
(vii) Suggest supplementary reading material (Section 1.7).

The reader eager to get started can skip this chapter without losing anything essential to understand the rest of the text.

1.1 The big picture through the OSI layering model

When we communicate using electronic devices, we produce streams of bits that typically go through various networks and are processed by devices from a variety of manufacturers. The system is very complex and there are a number of things that can go wrong. It is amazing that we can communicate as easily and reliably as we do. This could hardly be possible without layering and standardization. The Open System Interconnection (OSI) layering model of Figure 1.1 describes a framework for the definition of data-flow protocols. In this section we use the OSI model to convey the basic idea of how modern communication networks deal with the key challenges, notably routing, flow control, reliability, privacy, and authenticity. For the sake of concreteness, let us take e-mailing as a sample activity. Computers use bytes (8 bits) or multiples thereof to represent letters. So the content of an e-mail

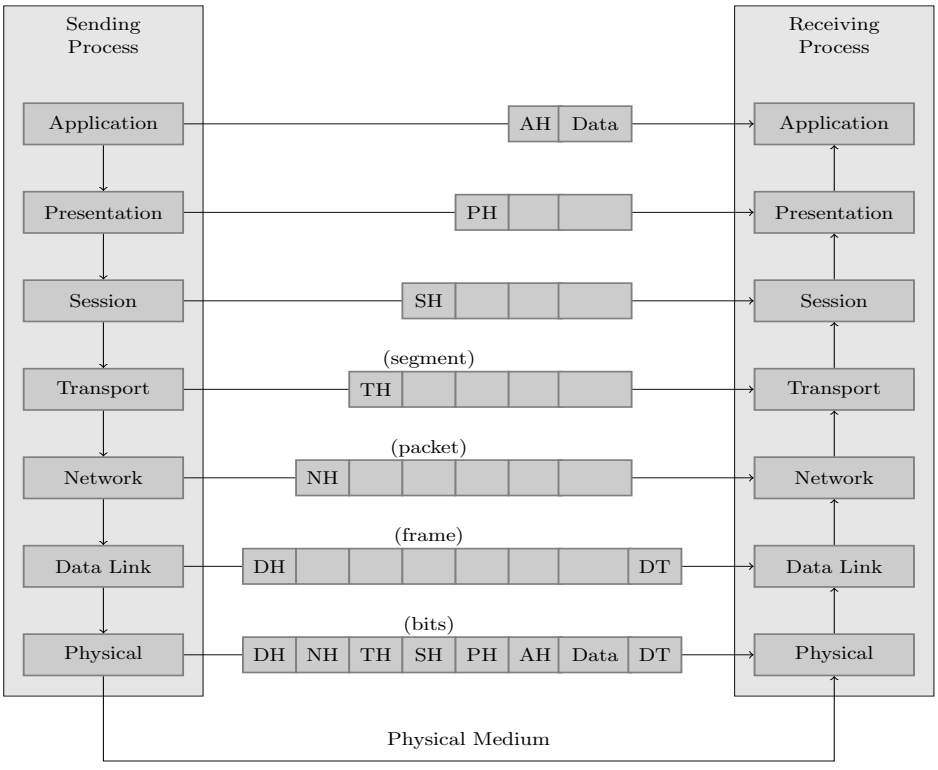

Figure 1.1. OSI layering model.

is represented by a stream of bytes. Received e-mails usually sit on a remote server. When we launch a program to read e-mail – hereafter referred to as the client – it checks into the server to see if there are new e-mails. It depends on the client's setting whether a new e-mail is automatically downloaded to the client or just a snippet is automatically downloaded until the rest is explicitly requested. The client tells the server what to do. For this to work, the server and the client not only need to be able to communicate the content of the mail message but they also need to talk to each other for the sake of coordination. This requires a protocol. If we use a dedicated program to do e-mail (as opposed to using a Web browser), the common protocols used for retrieving e-mail are the IMAP (Internet Message Access Protocol) and the POP (Post Office Protocol), whereas for sending e-mail it is common to use the SMTP (Simple Mail Transfer Protocol).

The idea of a protocol is not specific to e-mail. Every application that uses the Internet needs a protocol to interact with a peer application. The OSI model reserves the *application layer* for programs (also called processes) that implement application-related protocols. In terms of data traffic, the protocol places a so-called *application header* (AH) in front of the bits produced by the application. The top arrow in Figure 1.1 indicates that the two application layers talk to each other as if they had a direct link.

Typically, there is no direct physical link between the two application layers. Instead, the communication between application layers goes through a shared network, which creates a number of challenges. To begin with, there is no guarantee of privacy for anything that goes through a shared network. Furthermore, networks carry data from many users and can get congested. Hence, if possible, the data should be compressed to reduce the traffic. Finally, there is no guarantee that the sending and the receiving computers represent letters the same way. Hence, the application header and the data need to be communicated by using a universal language. The *presentation layer* handles the encryption, the compression, and the translation to/from a universal language. The presentation layer also needs a protocol to talk to the peer presentation layer at the destination. The protocol is implemented by means of the *presentation header* (PH).

For the presentation layers to talk to each other, we need to make sure that the two hosting computers are connected. Establishing, maintaining, and ending communication between physical devices is the job of the *session layer*. The session layer also manages access rights. Like the other layers, the session layer uses a protocol to interact with the peer session layer. The protocol is implemented by means of the *session header* (SH).

The layers we have discussed so far would suffice if all the machines of interest were connected by a direct and reliable link. In reality, links are not always reliable. Making sure that from an end-to-end point of view the link appears reliable is one of the tasks of the *transport layer*. By means of parity check bits, the transport layer verifies that the communication is error-free and if not, it requests retransmission. The transport layer has a number of other functions, not all of which are necessarily required in any given network. The transport layer can break long sequences into shorter ones or it can multiplex several sessions between the same two machines into a single one. It also provides flow control by queueing up data if the network is congested or if the receiving end cannot absorb it sufficiently fast. The transport layer uses the *transport header* (TH) to communicate with the peer layer. The transport header followed by the data handed down by the session layer is called a segment.

Now assume that there are intermediate nodes between the peer processes of the transport layer. In this case, the *network layer* provides the routing service. Unlike the above layers, which operate on an end-to-end basis, the network layer and the layers below have a process also at intermediate nodes. The protocol of the network layer is implemented in the *network header* (NH). The network header contains, among other things, the source and the destination address. The network header followed by the segment (of the transport layer) is called a *packet*.

The next layer is the *data link* (DL) layer. Unlike the other layers, the DL puts a header at the beginning and a trailer at the end of each packet handed down by the network layer. The result is called a *frame*. Some of the overhead bits are parity-check bits meant to determine if errors have occurred in the link between nodes. If the DL detects errors, it might ask to retransmit or drop the frame altogether. If it drops the frame, it is up to the transport layer, which operates on an end-to-end basis, to request retransmission.

The *physical layer* – the subject of this text – is the bottom layer of the OSI stack. The physical layer creates a more-or-less reliable "bit pipe" out of the physical channel between two nodes. It does so by means of a transmitter/receiver pair, called *modem*,[1] on each side of the physical channel. We will learn that the physical-layer designer can trade reliability for complexity and delay.

In summary, the OSI model has the following characteristics. Although the actual data transmission is vertical, each layer is programmed as if the transmission were horizontal. For a process, whatever is not part of its own header is considered as being actual data. In particular, a process makes no distinction between the headers of the higher layers and the actual data segment. For instance, the presentation layer translates, compresses, and encrypts whatever it receives from the application layer, attaches the PH, and sends the result to its peer presentation layer. The peer in turn reads and removes the PH and decrypts, decompresses, and translates the data which is then passed to the application layer. What the application layer receives is identical to what the peer application layer has sent up to a possible language translation. The DL inserts a trailer in addition to a header. All layers, except the transport and the DL layer, assume that the communication to the peer layer is error-free. If it can, the DL layer provides reliability between successive nodes. Even if the reliability between successive nodes is guaranteed, nodes might drop packets due to queueing overflow. The transport layer, which operates at the end-to-end level, detects missing segments and requests retransmission.

It should be clear that a layering approach drastically simplifies the tasks of designing and deploying communication infrastructures. For instance, a programmer can test the application layer protocol with both applications running on the same computer – thus bypassing all networking problems. Likewise, a physical-layer specialist can test a modem on point-to-point links, also disregarding networking issues. Each of the tasks of compressing, providing reliability, privacy, authenticity, routing, flow control, and physical-layer communication requires specific knowledge. Thanks to the layering approach, each task can be accomplished by people specialized in their respective domain. Similarly, equipment from different manufacturers work together, as long as they respect the protocols.

The OSI architecture is a generic model that does not prescribe a specific protocol. The Internet uses the TCP/IP protocol stack, which is essentially compatible with the OSI architecture but uses five instead of seven layers [4]. The reduction is mainly obtained by combining the OSI application, presentation, and session layers into a single layer called the application layer. The transport layer

[1] *Modem* is the result of contracting the terms *mod*ulator and *dem*odulator. In analog modulation, such as frequency modulation (FM) and amplitude modulation (AM), the source signal modulates a parameter of a high-frequency oscillation, called the carrier signal. In AM it modulates the carrier's amplitude and in FM it modulates the carrier's frequency. The modulated signal can be transmitted over the air and in the absence of noise (which is never the case) the demodulator at the receiver reconstructs an exact copy of the source signal. In practice, due to noise, the reconstruction only approximates the source signal. Although modulation and demodulation are misnomers in digital communication, the term modem has remained in use.

is implemented either by the *Transmission Control Protocol* (TCP) or by the *User Datagram Protocol* (UDP). The network layer implements addressing and routing via the *Internet Protocol* (IP). The DL and the physical layers complete the stack.

1.2 The topic of this text and some historical perspective

This text is about the theory that governs the physical-layer design (bottom layer in Figure 1.1), referred to as communication theory. Of course, other layers are about communication as well, and the reader might wonder why communication theory is not about all the layers. The terminology became established in the early days, prior to the OSI model, when communication was mainly point-to-point. Rather than including the other layers as they became part of the big picture, communication theory remained "faithful" to its original domain. The reason is most likely due to the dissimilarity between the body of knowledge needed to reason about the objectives of different layers. To gain some historical perspective, we summarize the key developments that have led to communication theory.

Electromagnetism was discovered in the 1820s by Ørsted and Ampère. The wireline telegraph was demonstrated by Henry and Morse in the 1830s. Maxwell's theory of electromagnetic fields, published in 1865, predicted that electromagnetic fields travel through space as waves, moving at the speed of light. In 1876, Bell invented the telephone. Around 1887, Hertz demonstrated the transmission and reception of the electromagnetic waves predicted by Maxwell. In 1895, Marconi built a wireless system capable of transmitting signals over more than 2 km. The invention of the vacuum-tube diode by Fleming in 1904 and of the vacuum-tube triode amplifier by Forest in 1906 enabled long-distance communication by wire and wireless. The push for sending many phone conversations over the same line led, in 1917, to the invention of the wave filter by Campbell.

The beginning of digital communication theory is associated with the work of Nyquist (1924) and Hartley (1928), both at Bell Laboratories. Quite generally, we communicate over a channel by choosing the value of one or more parameters of a carrier signal. Intuitively, the more parameters we can choose independently, the more information we can send with one signal, provided that the receiver is capable of determining the value of those parameters.

A good analogy to understand the relationship between a signal and its parameters is obtained by comparing a signal with a point in a three-dimensional (3D) space. A point in 3D space is completely specified by the three coordinates of the point with respect to a Cartesian coordinate system. Similarly, a signal can be described by a number n of parameters with respect to an appropriately chosen reference system called the orthonormal basis. If we choose each coordinate as a function of a certain number of bits, the more coordinates n we have the more bits we can convey with one signal.

Nyquist realized that if the signal has to be confined to a specified time interval of duration T seconds (e.g. the duration of the communication) and frequency interval of width B Hz (e.g. the channel bandwidth), then the integer n can be

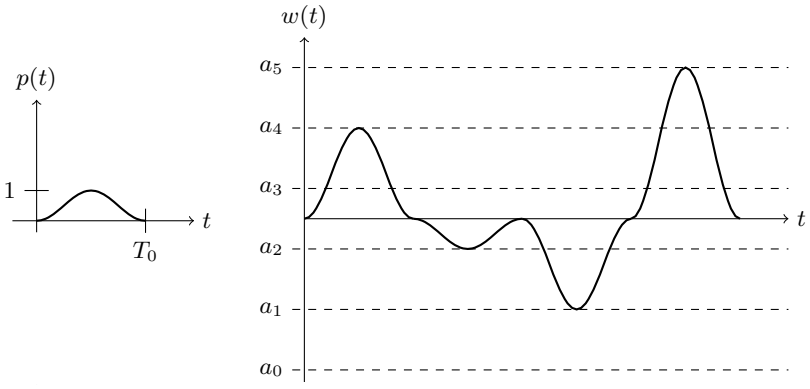

Figure 1.2. Information-carrying pulse train. It is the scaling factor of each pulse, called symbol, that carries information. In this example, the symbols take value in $\{a_0, a_1, a_2, a_3, a_4, a_5\}$.

chosen to be close to ηBT, where η is some positive number that depends on the definition of duration and bandwidth. A good value is $\eta = 2$.

As an example, consider Figure 1.2. On the left of the figure is a pulse $p(t)$ that we use as a building block for the communication signal.[2] On the right is an example of a pulse train of the form $w(t) = \sum_{i=0}^{3} s_i p(t - iT_0)$, obtained from shifted and scaled replicas of $p(t)$. We communicate by scaling the pulse replica $p(t - iT_0)$ by the information-carrying *symbol* s_i. If we could substitute $p(t)$ with a narrower pulse, we could fit more such pulses in a given time interval and therefore we could send more information-carrying symbols. But a narrower pulse uses more bandwidth. Hence there is a limit to the pulse width. For a given pulse width, there is a limit to the number of pulses that we can pack in a given time interval if we want the receiver to be able to retrieve the symbol sequence from the received pulse train. Nyquist's result implies that we can fit essentially $2BT$ non-interfering pulses in a time interval of T seconds if the bandwidth is not to exceed B Hz.

In trying to determine the maximum number of bits that can be conveyed with one signal, Hartley introduced two constraints that make good engineering sense. First, in a practical realization, the symbols cannot take arbitrarily large values in $\mathbb{R}$ (the set of real numbers). Second, the receiver cannot estimate a symbol with infinite precision. This suggests that, to avoid errors, symbols should take values in a discrete subset of some interval $[-A, A]$. If $\pm\Delta$ is the receiver's precision in determining the amplitude of a pulse, then symbols should take a value in some alphabet $\{a_0, a_1, \ldots, a_{m-1}\} \subset [-A, A]$ such that $|a_i - a_j| \geq 2\Delta$ when $i \neq j$. This implies that the alphabet size can be at most $m = 1 + \frac{A}{\Delta}$ (see Figure 1.3).

There are m^n distinct n-length sequences that can be formed with symbols taken from an alphabet of size m. Now suppose that we want to communicate a sequence

[2] A pulse is not necessarily rectangular. In fact, we do not communicate via rectangular pulses because they use too much bandwidth.

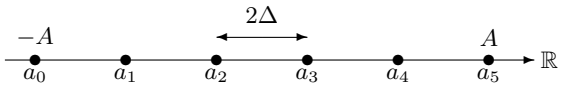

Figure 1.3. Symbol alphabet of size $m = 6$.

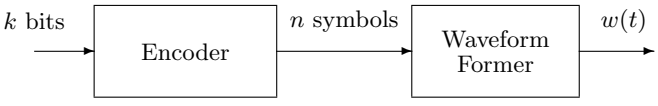

Figure 1.4. Transmitter.

of k bits. There are 2^k distinct such sequences and each such sequence should be mapped into a distinct symbol sequence (see Figure 1.4). This implies

$$2^k \leq m^n. \tag{1.1}$$

EXAMPLE 1.1 *There are $2^4 = 16$ distinct binary sequences of length $k = 4$ and there are $4^2 = 16$ distinct symbol sequences of length $n = 2$ with symbols taking value in an alphabet of size $m = 4$. Hence we can associate a distinct length-2 symbol sequence to each length-4 bit sequence. The following is an example with symbols taken from the alphabet $\{a_0, a_1, a_2, a_3\}$.*

bit sequence	*symbol sequence*
0000	a_0a_0
0001	a_0a_1
0010	a_0a_2
0011	a_0a_3
0100	a_1a_0
⋮	⋮
1111	a_3a_3

□

Inserting $m = 1 + \frac{A}{\Delta}$ and $n = 2BT$ in (1.1) and solving for $\frac{k}{T}$ yields

$$\frac{k}{T} \leq 2B \log_2 \left(1 + \frac{A}{\Delta}\right) \tag{1.2}$$

as the highest possible rate in bits per second that can be achieved reliably with bandwidth B, symbol amplitudes within $\pm A$, and receiver accuracy $\pm\Delta$.

Unfortunately, (1.2) does not provide a fundamental limit to the bit rate, because there is no fundamental limit to how small Δ can be made.

The missing ingredient in Hartley's calculation was the noise. In 1926 Johnson, also at Bell Labs, realized that every conductor is affected by thermal noise. The idea that the received signal should be modeled as the sum of the transmitted signal plus noise became prevalent through the work of Wiener (1942). Clearly the noise

prevents the receiver from retrieving the symbols' values with infinite precision, which is the effect that Hartley wanted to capture with the introduction of Δ, but unfortunately there is no way to choose Δ as a function of the noise. In fact, in the presence of thermal noise, error-free communication becomes impossible. (But we can make the error probability as small as desired.)

Prior to the publication of Shannon's revolutionary 1948 paper, the common belief was that the error probability induced by the noise could be reduced only by increasing the signal's power (e.g. by increasing A in the example of Figure 1.3) or by reducing the bit rate (e.g. by transmitting the same bit multiple times). Shannon proved that the noise can set a limit to the number of bits per second that can be transmitted reliably, but as long as we communicate below that limit, the error probability can be made as small as desired without modifying the signal's power and bandwidth. The limit to the bit rate is called *channel capacity*. For the setup of interest to us it is the right-hand side of

$$\frac{k}{T} \leq B \log_2 \left(1 + \frac{P}{N_0 B}\right), \tag{1.3}$$

where P is the transmitted signal's power and $N_0/2$ is the power spectral density of the noise (assumed to be white and Gaussian). If the bit rate of a system is above channel capacity then, no matter how clever the design, the error probability is above a certain value. The theory that leads to (1.3) is far more subtle and far more beautiful than the arguments leading to (1.2); yet, the two expressions are strikingly similar.

What we mentioned here is only a special case of a general formula derived by Shannon to compute the capacity of a broad class of channels. As he did for channels, Shannon also posed and answered fundamental questions about sources. For the purpose of this text, there are two lessons that we should retain about sources. (1) The essence of a source is its randomness. If a listener knew exactly what a speaker is about to say, there would be no need to listen. Hence a source should be modeled by a random variable (or a sequence thereof). In line with the topic of this text, we assume that the source is digital, meaning that the random variable takes values in a discrete set. (See Appendix 1.8 for a brief summary of various kind of sources.) (2) For every such source, there exists a source encoder that converts the source output into the shortest (in average) binary string and a source decoder that reconstructs the source output from the encoder output. The encoder output, for which no further compression is possible, has the same statistic as a sequence of unbiased coin flips, i.e. it is a sequence of independent and uniformly distributed bits. Clearly, we can minimize the storage and/or communicate more efficiently if we compress the source into the shortest binary string. In this text, we are not concerned with source coding but, for the above-mentioned reasons, we model the source as a generator of independent and uniformly distributed bits.

Like many of the inventors mentioned above, Shannon worked at Bell Labs. His work appeared one year after the invention of the solid-state transistor, by Bardeen, Brattain, and Shockley, also at Bell Labs. Figure 1.5 summarizes the various milestones.

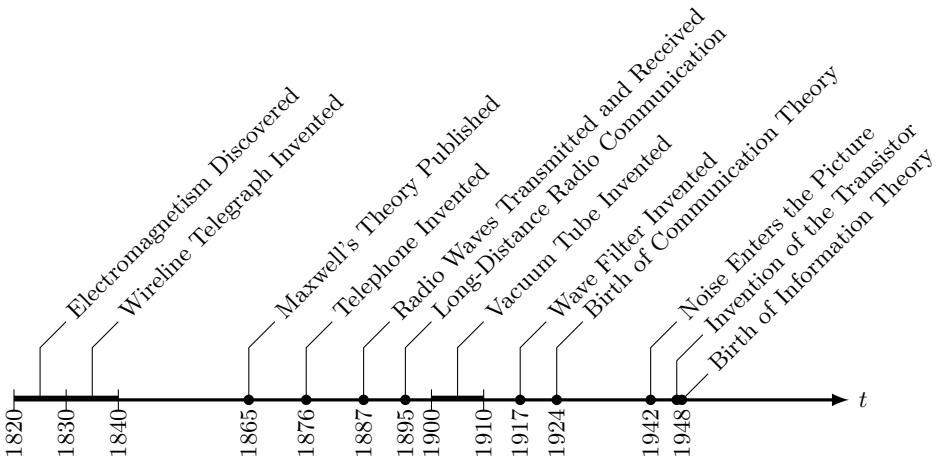

Figure 1.5. Technical milestones leading up to information theory.

Information theory, which is mainly concerned with discerning between what can be done and what cannot, regardless of complexity, led to coding theory – a field mainly concerned with implementable ways to achieve what information theory proves as achievable. In particular, Shannon's 1948 paper triggered a race for finding implementable ways to communicate reliably (i.e. with low error probability) at rates close to the channel capacity. Coding theory has produced many beautiful results: too many to summarize here; yet, approaching channel capacity in wireline and wireless communication was out of reach until the discovery of turbo codes by Berrou, Glavieux, and Thitimajshima in 1993. They were the first to demonstrate a method that could be used to communicate reliably over wireline and wireless channels, with less than 1 dB of power above the theoretical limit. As of today, the most powerful codes for wireless and wireline communication are the low-density parity-check (LDPC) codes that were invented in 1960 by Gallager in his doctoral dissertation at MIT and reinvented in 1996 by MacKay and Neal. When first invented, the codes did not receive much attention because at that time their extraordinary performance could not be demonstrated for lack of appropriate analytical and simulation tools. What also played a role is that the mapping of LDPC codes into hardware requires many connections, which was a problem before the advent of VLSI (very-large-scale integration) technology in the early 1980s.

1.3 Problem formulation and preview

Our focus is on the system aspects of digital point-to-point communication. By the term *system aspect* we mean that we remain at the level of building blocks rather than going into electronic details; *digital* means that the message is taken from a finite set of possibilities; and we restrict ourselves to *point-to-point* communication as it constitutes the building block of all communication systems.

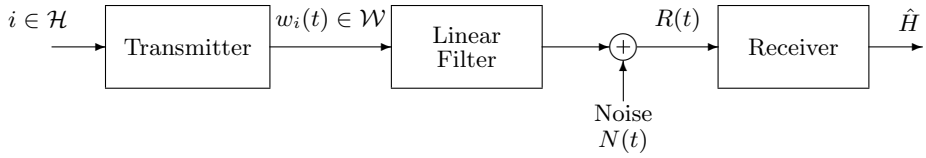

Figure 1.6. Basic point-to-point communication system over a bandlimited Gaussian channel.

Digital communication is a rather unique field in engineering in which theoretical ideas from probability theory, stochastic processes, linear algebra, and Fourier analysis have had an extraordinary impact on actual system design. The mathematically inclined will appreciate how these theories seem to be so appropriate to solve the problems we will encounter.

Our main target is to acquire a solid understanding about how to communicate through a channel, modeled as depicted in Figure 1.6. The source chooses a message represented in the figure by the index i, which is the realization of a random variable H that takes values in some finite alphabet $\mathcal{H} = \{0, 1, \ldots, m-1\}$. As already mentioned, in reality, the message is represented by a sequence of bits, but for notational convenience it is often easier to label each sequence with an index and use the index to represent the message. The transmitter maps a message i into a signal $w_i(t) \in \mathcal{W}$, where $\mathcal{W}$ and $\mathcal{H}$ have the same cardinality. The channel filters the signal and adds Gaussian noise $N(t)$. The receiver's task is to guess the message based on the channel output $R(t)$. So $\hat{H}$ is the receiver's guess of H. (Owing to the random behavior of the noise, $\hat{H}$ is a random variable even under the condition that $H = i$.)

In a typical scenario, the channel is given and the communication engineer designs the transmitter/receiver pair, taking into account objectives and constraints. The objective could be to maximize the number of bits per second being communicated, while keeping the error probability below some threshold. The constraints could be expressed in terms of the signal's power and bandwidth.

The noise added by the channel is Gaussian because it represents the contribution of various noise sources.[3] The filter has both a physical and a conceptual justification. The conceptual justification stems from the fact that most wireless communication systems are subject to a license that dictates, among other things, the frequency band that the signal is allowed to occupy. A convenient way for the system designer to deal with this constraint is to assume that the channel contains an ideal filter that blocks everything outside the intended band. The physical reason has to do with the observation that the signal emitted from the transmitting antenna typically encounters obstacles that create reflections and scattering. Hence the receiving antenna might capture the superposition of a number of delayed and attenuated replicas of the transmitted signal (plus noise). It is a straightforward

[3] Individual noise sources do not necessarily have Gaussian statistics. However, due to the central limit theorem, their aggregate contribution is often quite well approximated by a Gaussian random process.