Transmission on a Serial Line

Norman Matloff

University of California at Davis

September 3, 2001

1 Overview

Consider two or more network entities, such as

(a) connecting one computer to another, as a link in a wide-area network

(b) connecting many terminals to one computer

(c) connecting one computer's modem to another computer's modem

(In the example (b), the "terminals" could take on various forms. In earlier decades, the word **terminal** would usually mean a keyboard-monitor combination whose sole purpose was for the user to communicate with the host computer. Since the terminal was not a computer in its own right and thus could not run programs itself, it was often called a "dumb" terminal. Today such a role is more often played by PCs, such that the user can not only communicate with the host computer but also perform his/her own local processing on the PC. However, a common application today which is similar to the "dumb terminal" setting of the past is a network of barcode scanners in a supermarket, all connected to a central computer which maintains prices and inventories for the items sold by the store.)

If just two entities are connected, as in examples (a) and (c) above, we call the line a **point-to-point** link. In example (b), we call the line a **multipoint** or **multidrop** link.

Our interest here will be a communications protocols on the link which have some tolerance for line noise and discrepancies between the clocks at the two ends of the link. Such protocol are divided into **asynchronous transmission** and **synchronous transmission**.

The main difference between the two approaches is that asynchronous transmissio deals with line-noise errors and clock discrepancies on a character-by-character basis, while synchronous protocols check errors and clocking on the basis of blocks of characters (or blocks of bits).

Here we have a single line, on which bits must be transmitted serially, that is one bit at a time. Note that some system must be decided up for coding 1s and 0s. We will assume the NRZ-L system, which in metal

wire means a low voltage for a 1 and a high voltage for a 0; in optical fibers, this would correspond to light-off for a 1 and light-on for a 0. If the serial line is being used for modem-to-modem communication, 1s and 0s will be coded with high- and low-pitched tones.

2 Asynchronous Transmission

This type of communication protocol is quite common. For example, the serial ports in your PC use this protocol.

2.1 Frame Format

The transmission of a single character will consist of the following, in the sequence given:

- A Start bit, consisting of a 0.
- The character itself, coded as k bits, where $5 \leq k \leq 8$. The value of k is fixed for a given transmitter/receiver pair. We will assume k = 8 from this point on.
- A Parity bit, set on either an even or odd basis, explained below. We will assume even parity.
- One or two Stop bits, consisting of 1s. We will assume one Stop bit.

Altogether, then the transmission of a character will consist of 1+8+1+1 = 11 bits. If upon completion of transmission of one character another character is to immediately follow, the Start bit of the latter will immediately follow the Stop bit of the former. Otherwise, the line will be in Idle state, i.e. a 1, until a new character is ready for transmission.

2.2 Clock Synchronization

After receiving a character (including the Parity bit), the electronics in the receiver are set up to constantly monitor the line, watching for 1-to-0 transitions. This explains the role of the Start and Stop bits:

- If the line had been idle before the character became ready for transmission, the 0 value in the Start bit will cause the 1-to-0 transition on the line.
- If this character immediately follows the preceding one, this one's Start bit will comprise a 1-to-0 transition from the preceding one's Stop bit.

The purpose of the Stop bit can be viewed as a 1-bit forced idle time. We need to have the line at 1 before the Start bit of a character, in order to get the 1-to-0 transition. This comes "naturally" if there is idle time preceding that character, but if immediately follows an earlier character, then that earlier character's Stop bit will serve as an artificial "idle" time.

When the 1-to-0 transition is detected, the receiver treats this as the start of a character. It waits half a bit time and then starts its clock, for synchronization purposes, setting it to 0.[1] The receiver will sample the line at the times at which the midpoints of the subsequent bits are on the line.[2]

Note that (under the design assumed here), if somehow the Stop bit gets corrupted, the receiver will ignore all succeeding bits until it sees a 1-to-0 transition. The point is that that transition is crucial, as it is used for clock synchronization.

To illustrate how this works, let us take as an example a line with a transmission rate of 10,000 bits per second. (Again, this is just an example.) That means that each bit has a duration of 100 microseconds (100 μs). It will then sample at time 150, the middle of the first Data bit, and record whatever value it finds on the line as the first Data bit. After that, it will sample every 100 μs, i.e. at times 250, 350 and so on, up to time 950, when it samples the Parity bit. (It really does not have to sample the Stop bit, since the only use, if any, for that bit is to set up detection of the start of the next character.)

Ostensibly the receiver's clock has the same rate as that of the transmitter, but of course no two clocks can be exactly the same. This presents a problem. Suppose, for instance, that the receiver clock is 7% fast, so that 100 μs on that clock corresponds to only 93 μs on the transmitter's clock. Then the receiver's first line sampling will at "real time" (at least from the transmitter's perspective) 139.5. This is not a problem, since that first Data bit will be on the line from times 100 to 200. Similarly, we will not have a problem with the second, third, fourth and fifth bits. But after that it becomes risky, and we will have a problem with at least the eighth bit. The bit will be sampled at time 790.5, but it does not even get on the line until time 800. In other words, when the receiver thinks it is sampling the eighth bit, it is actually picking up the seventh!

So, a discrepancy in clock rates of 7% would be a disaster. However, an analysis similar to that above shows that a difference of under 5% would not produce problems.

So, the protocol does achieve its goal of coping with discrepancies in the clock rates, as long as they are not too large. By the way, you can see from this why this protocol handles this problem on a byte-by-byte basis. If for example it instead used two bytes as the basic unit, surrounded by Start, Parity and Stop bits, the amount of tolerable clock discrepancy would be even smaller. Since the receiver clock is restarted each time a Start bit is encountered, and thus resynchronized with the sender clock, we do not have to worry about the two clocks drifting apart by too much during a data transmission.

2.3 Error Checking

The transmitter sets the Parity bit to 1 or 0, whichever makes the total number of 1 bits among the eight Data bits and one Parity bit an even number. So, if the Data byte is, say, 01000011, then the Parity bit will be set to 1.

The receiver will then count the number of 1s it receives among those nine bits. If the count is odd, then the receiver knows that at least one bit was received in error (probably due to line noise). Note, though, that if the number of errors is even, the receiver will not sense that anything is wrong, because the number of 1s it receives will still be even.

[1] So the term **asynchronous transmission** is a misnomer.

[2] More precisely, we should say that it will sample the line at the times which it *thinks* are the midpoints.

2.4 Evaluation

The asynchronous transmission protocol has the virtue of being simple. However, it suffers from high overhead—2/11 of the bandwidth of the line is being used for synchronization, not data. Moreover, the probability of undetected errors is not as low as we would like it to be.