Baseline Wandering and Self-Synchronization (Clock Recovery)

Consider the Polar NRZ-L line coding technique. The problem with NRZ is that a sequence of several consecutive 1s means that the signal stays high on the link for an extended period of time; similarly, several consecutive 0s means that the signal stays low for a long time.

There are two fundamental problems caused by long strings of 1s or 0s.

The first is that it leads to a situation known as *baseline wander*. Specifically, the receiver keeps an average of the signal it has seen so far and then uses this average to distinguish between low and high signals. Whenever the signal is significantly lower than this average, the receiver concludes that it has just seen a 0; likewise, a signal that is significantly higher than the average is interpreted to be a 1. The problem, of course, is that too many consecutive 1s or 0s cause this average to change, making it more difficult to detect a significant change in the signal.

The second problem is that frequent transitions from high to low and *vice versa* are necessary to enable *clock recovery (i.e., self-synchronization)*. Intuitively, the clock recovery problem is that both the encoding and the decoding processes are driven by a clock—every clock cycle the sender transmits a bit and the receiver recovers a bit. The sender's and the receiver's clocks have to be precisely synchronized in order for the receiver to recover the same bits the sender transmits. If the receiver's clock is even slightly faster or slower than the sender's clock, then it does not correctly decode the signal. You could imagine sending the clock to the receiver over a separate wire, but this is typically avoided because it makes the cost of cabling twice as high. So, instead, the receiver derives the clock from the received signal—the clock recovery process. Whenever the signal changes, such as on a transition from 1 to 0 or from 0 to 1, then the receiver knows it is at a clock cycle boundary, and it can resynchronize itself. However, a long period of time without such a transition leads to clock drift. Thus, clock recovery depends on having lots of transitions in the signal, no matter what data is being sent.

One approach that addresses this problem, called *non-return to zero inverted* (NRZ-I), has the sender make a transition from the current signal to encode a 1 and stay at the current signal to encode a 0. This solves the problem of consecutive 1s, but obviously does nothing for consecutive 0s. NRZI is illustrated in Figure 1. An alternative, called *Manchester encoding*, does a more explicit job of merging the clock with the signal by transmitting the exclusive OR of the NRZ-encoded data and the clock. (Think of the local clock as an internal signal that alternates from low to high; a low/high pair is considered one clock cycle.) The Manchester encoding is also illustrated in Figure 1. Observe that the Manchester encoding results in 0 being encoded as a low-to-high transition and 1 being encoded as a high-to-low transition. Because both 0s and 1s result in a transition to the signal, the clock can be effectively recovered at the receiver. (There is also a variant of the Manchester encoding, called *Differential Manchester*, in which a 1 is encoded with the first half of the signal equal to the last half of the previous bit's signal and a 0 is encoded with the first half of the signal opposite to the last half of the previous bit's signal.)

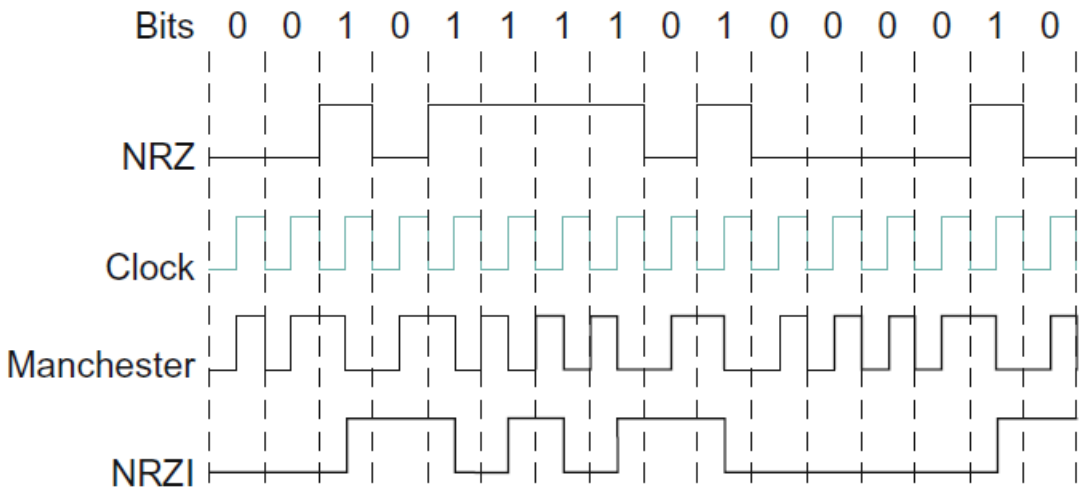

Figure 1: Different encoding schemes.

Reference

Larry L. Peterson and Bruce S. Davie, Computer Networks: A Systems Approach, 5th Ed, 2012 (Chapter 2).