Lecture 16: Wireless bit-rate adaptation

Anirudh Sivaraman

2018/12/17

This lecture, we'll continue with the physical layer, but applied to wireless networks instead of wired networks. In particular, we'll talk about one problem that occurs in the wireless physical layer: wireless bit-rate adaptation. We'll start by defining bit-rate adaptation. Then we'll talk about bit-rate adaptation in an idealized model of the wireless environment. We'll then discuss bit-rate adaptation in practice by describing the SampleRate bit-rate adaptation algorithm [1], one of the earliest bit-rate adaptation algorithms.

1 The problem of bit-rate selection

Recall from the previous lecture that we translate digital information (bits) into analog signals (voltages) prior to transmitting these signals using modulation. During this translation process we are constrained by a finite range of voltages. More precisely, we are constrained by a finite power requirement. However, power is proportional to the square of the voltage, and hence a finite power constraint translates into a finite voltage constraint as well.

We would like to pack in as many different voltage levels within this range because each voltage level (formally called a symbol) can be used to transmit a distinct digital value. Hence, the more the number of symbols within the finite voltage range, the more the number of distinct values, and hence the more the number of bits we can transmit in the time that it takes to transmit a symbol over the physical medium, i.e., the symbol time. This *symbol time* is dictated by electrical engineering concerns and we'll assume it's some constant for this lecture.

The number of bits that can be transmitted within a symbol time is just the logarithm (to the base 2) of the number of distinct symbols within the voltage range. The bit rate of a wireless channel is defined as the number of bits that you can send within a symbol divided by the symbol time. As a concrete example, let's say we can transmit a symbol every microsecond, which is in the ballpark of WiFi physical layers these days. The symbol time is then 1 microsecond. Let's also say you have a finite 5 V range and you pack in 16 voltage levels (or symbols) into this range: $\frac{5}{16}$ V, $\frac{10}{16}$ V, $\frac{15}{16}$ V, and so on until 5 V. Because there are 16 distinct values, you could represent up to 4 bits using each of these symbols. Hence, the bits per symbol is 4, and the bit rate is 4 bits every microsecond or 4 Mbit/s.

The goal of bit-rate adaptation is to maximize the rate at which bits can be sent and successfully received (i.e., without bit flips, which we describe below).

2 A simplified model for bit-rate adaptation

What constrains the number of symbols we can pack into a voltage range? It's noise, or more precisely the ratio of the signal to the noise (SNR), as we discussed last lecture. If the noise in the physical layer is high relative to the signal, it's possible that a symbol transmitted at the sender gets perturbed by enough noise causing it to "transform" into another symbol at the receiver.

This transformation into another symbol manifests itself as a transformation in the bits representing the symbol. For instance, if there were 16 symbols within a 5 V range, you could represent up to 4 bits using each symbol. Then, if you transmitted the bits 0001 (i.e, symbol value 1), it would correspond to the second symbol

($\frac{10}{16}$ V).[1] If this symbol got perturbed by enough noise, it could transform into the previous symbol ($\frac{5}{16}$ V), which corresponds to the bits 0000 (i.e., symbol value 0). In other words, the bits 0001 have been transformed into 0000, and the last bit 1 has been flipped into the bit 0.

The probability of a bit flip is called the bit error rate (BER) to reflect the fact that a bit has been erroneously received—either a 0 received as a 1 or a 1 received as a 0. The BER depends on two attributes: (1) if the packing configuration (i.e., number of symbols within a voltage range) is held constant, a low SNR causes more bit flips because the large noise levels can transform one symbol into another, (2) if the SNR is held constant, a tighter packing configuration causes more bit flips because symbols are more closely packed within the voltage range giving much lower "margin for error."

Once we take into account the BER (which is a function of the SNR and the packing configuration) and the bits per symbol (which is function of the packing configuration alone), the effective bit rate for a given packing configuration and SNR is:

$$\frac{(1 - BER) * (bitspersymbol)}{symboltime} \tag{1}$$

The 1 - BER term reflects the probability that a bit is delivered correctly.

This presents us with a tradeoff at any SNR: you could use a tighter packing, which gives you a higher bits per symbol, but there is the risk that this increases the BER, which decreases the overall the bit rate. Or you could use a looser packing, which gives you a lower bits per symbol, but decreases the BER as well.

This tradeoff is at the heart of bit-rate adaptation. The job of bit-rate adaptation is to pick the right packing configuration at a given SNR. To pick the right packing configuration, it's ultimately the product of the two terms (1 - BER) and *bitspersymbol* that matters—not either one in isolation.

3 The SampleRate algorithm

We'll now describe one example of bit-rate adaptation in practice: the SampleRate algorithm [1]. The first deviation from the simplified model described above is that it is hard to estimate BER as a function of the SNR in practice. This is for two reasons. First, while there are mathematical equations that map SNR to BER, these equations hold in idealized models of the physical layer like a single sender-receiver pair. These idealizations don't typically hold in practice where there are multiple senders and receivers with walls and other obstacles between them. Second, it is hard to precisely measure SNR on most WiFi cards today.

But the simplified model from the previous section points to one interesting observation: it is not the BER itself that matters, nor is it the bits per symbol of a particular packing configuration. It's rather their product that matters because it reflects the actual bit rate of the physical layer. Hence, as long as there is a way to measure the *physical-layer throughput* after accounting for bit errors at any given packing configuration, it's sufficient for bit-rate adaptation. This is because the physical-layer throughput is a lower bound for the bit rate, i.e., the bit rate can only ever be higher than the physical-layer throughput. With this measurement of physical-layer throughput, we should be able to experiment with different packing configurations and pick the configuration that empirically maximizes the physical-layer throughput. In essence, we can treat the wireless channel as a blackbox that takes packing configurations as inputs and produces physical-layer throughput values as outputs; the goal is to maximize physical-layer throughput.

The SampleRate algorithm, which runs independently for every sender-receiver pair, codifies these insights. It performs bit-rate adaptation for every pair by carefully estimating throughput by considering as many factors as possible. Here's how the algorithm works.

1. Send data at the highest packing configuration, i.e., the configuration with the most number of bits per symbol.

2. Stop using this packing configuration after 4 successive packets were not delivered to the receiver at that configuration.[2] The failure of a packet delivery is determined in WiFi through the absence of an ACK after

[1] Symbol values start from 0 (0000) and go until 15 (1111). Hence, the first symbol has symbol value 0, the second symbol has symbol value 1, and so on.

[2] There is nothing sacred about the number 4 here. It's just a constant that is chosen empirically.

a certain timeout. Repeat this until you find a packing configuration where you don't see 4 successive packet transmission failures.

3. Every 10 data packets,[3] pick a random packing configuration from an eligible set of packing configurations. The eligible set is the set of configurations that have an average transmission time lower than the average transmission time for the current packing configuration. Once a random configuration is picked from the eligible set, switch to this newly picked packing configuration.

The most interesting part of SampleRate is how the average transmission time (i.e., the inverse of the throughput we mentioned in the previous paragraph) is calculated. The average transmission time at a particular packing configuration is calculated by using the packet's length, the bit rate achievable at that packing configuration assuming a BER of 0, the time spent backing off according to the WiFi CSMA/CA protocol, and the number of retries required to successfully transmit the packet from the sender to the receiver. The transmission time of a particular packet is roughly:[4]

$$backofftime + (numberofretries) * ((packetlength)/(bitrate)) \quad (2)$$

To compute the average transmission time at a particular packing configuration, SampleRate averages the individual transmission times of all packets sent and received successfully at that configuration over the last 10 seconds.

The key idea in SampleRate is the average transmission time computation. This computation handles as many factors as possible when computing a true *end-to-end* version of the physical layer throughput, e.g., the possibility that a packet may be lost, the possibility that the WiFi user might spend some time backing off, and so on. Measuring performance end-to-end is important because the constituent factors in end-to-end performance (BER, bits per symbol) are less important than the final end-to-end performance itself. Further, in practice, it is hard to model the dependence of end-to-end performance (in this case throughput) on the constituent factors. Given that it's hard to model, why not measure it directly instead?

References

[1] John Bicket. Bit-rate Selection in Wireless Networks. Master's thesis, Massachusetts Institute of Technology, February 2005.

[3] Again, there is nothing sacred about the number 10 here.

[4] We are leaving out some WiFi technicalities. For the precise version, look at equation 5.1 in the SampleRate thesis [1].