Randomized Communication Protocols

Max von Hippel

April 6, 2021

- All players share a random string r.
- $R(f) := E[|C(f)|]$ is the *expected* # bits communicated.
- A randomized communication protocol is allowed to output the wrong answer at most $1/3$ of the time.

The "public coin" scenario is where you assume r is known ahead-of-time and does not need to be communicated. The "private coin" scenario is where someone computes r and shares it with the other players, so it contributes to the size of the protocol.

It turns out randomized protocols can be much faster, e.g.,

$$C(=) \geq n$$

but

$$R(=) \in \mathcal{O}(log n) \text{ with a private coin, or}$$
$$\in \mathcal{O}(1) \text{ with a public coin}$$

The protocol with a private coin turns out to be ModCoin, where

1. Alice chooses $\text{Coin} \in \mathbb{Z}_{2n}$ randomly;
2. Alice sends $\langle x \bmod \text{Coin}, \text{Coin} \rangle$ to Bob;
3. Bob sends 1 iff $x \bmod \text{Coin} = y \bmod \text{Coin}$ else 0, back to Alice.

- If $x = y$ the protocol is correct with probability 1.
- If $x \neq y$ the protocol is correct with probability $P[x \bmod \text{Coin} \neq y \bmod \text{Coin}]$ which turns out to exceed $1/2$.

The protocol with a public coin turns out to be Checksum, which

- if $x = y$ is correct with probability 1, and
- if $x \neq y$ is correct with probability $1/2$.

So if you just do this a second time to confirm $x \neq y$ you get $1/4$ error rate and you are done.

More details are not available in the book but a nice tutorial can be found here: https://www.cs.toronto.edu/~toni/Courses/CommComplexity2/Lectures/lecture1.pdf.