Nanopayments: When I Didn't Know Who to Trust, I Turned to Probability

Dr. Chloe I. Avery

September 17, 2024

Contents

1 Introduction/Disclaimer

How much of your money would you trust an internet stranger with? If you're at all like me, your answer is very little. Sure, maybe I'm okay with running the risk of losing pennies or fractions of pennies in order to give me access to all of the incredible, interesting, and useful services this new age of the internet has to offer. But honestly, much more than that and it is just not a risk I am willing to take. Maybe I'm being too careful, but I've also been scammed or witnessed scams too many times to count[1].

This paper describes probabalistic Nanopayments[2], a payment system that allows users to send and receive tiny amounts of value while minimizing transaction costs. Because yes, those gas fees make me want to cry too. Probabalistic Nanopayments were first described by researchers at Orchid in [1]. A note: while I do work for Orchid now, this research pre-dates me, so all of the credit for these very cool ideas goes to the authors of [1]. The main idea of probabalistic Nanopayments is this: instead of actually sending a small amount of value, the client sends what we call a "lottery ticket" with an expected value of the intended payment. These only pay out when the recipient receives a winning ticket. For example, imagine if instead of giving you a penny, I gave you a 1 in 100 chance of receiving a dollar, and did so in a way that was provably fair- meaning there is no way I could trick you into taking a ticket I know is a losing ticket, and there's no way you could trick me into giving you a ticket you know is a winner. Since they are provably fair, if I was paying you repeatedly in these 1 in 100 chances of winning a dollar, you might be willing to treat each one as if I was paying you a penny. We'll get into the math behind why sending these "lottery tickets" is equivalent[3] to repeatedly sending small payments in Section 4. I will note here that the key is this phrase "repeatedly sending small payments". I imagine that if I gave you a sandwich, and you gave me

[1] Okay, not *uncountably* many times. But, like, a lot.

[2] The name "Nanopayments" is a nod to the scheme MICROPAY1, outlined in [2], which motivated some concepts in the Nanopayments scheme.

[3] Alright you caught me. It's not *technically* equivalent, though the more payments you send, the closer it is to equivalent. So I guess we could say "it is equivalent in the limit". See Section 3 for what I mean by this.

a lottery ticket, one of us would end up pretty disappointed. Either the lottery ticket is a winner, in which case you are definitely going to feel like you overpaid, or it is a losing ticket, in which case I am going to feel like I gave you a sandwich for free. This idea works a lot better if, say, I have a sandwich shop, and my customers pay me in lottery tickets per bite. Because we are exchanging lots and lots of these lottery tickets, the payment eventually evens out. And this way, my customers are able to eat exactly as much as they want- no still being hungry, no leftovers, and I am getting paid in a way that minimizes the amount of transaction fees I have to pay, resulting in higher profit margins.

So many virtual services rely on the subscription model- an up-front payment for use of a service for a specified period of time, often on the timescale of a month, and this subscription often renews at the end of this time period. This kind of a payment model can alienate potential users, including those who either just want to try the service and those who don't intend to use the service as often as the subscription cost would suggest. Probabalistic Nanopayments not only minimize the need for trust in a virtual transaction, but also open up a whole world of pay-as-you-go possibilities. Imagine, for example, if instead of having subscriptions to multiple different streaming services, you could have a single account and only pay for exactly what you watched. Nanopayments make these types of payment schemes possible!

2 Trust Bounds

In a game of trust, who makes the first move? My background is in mathematics, and in the mathematical field of Game Theory, the question of who goes first is often a huge consideration and can determine which player actually has a winning strategy. Turn order can be just as important in a virtual transaction. If first a service is provided, then a payment is made, the provider runs the risk of not being paid at all and unintentionally providing service for free. If a payment is made prior to a service being provided, then the client is the one to assume the risk, which could be receiving no service at all, or even receiving such poor quality of service that one would wish they hadn't received service in the first place.

We call the maximum amount of value that a user is willing to risk losing in a virtual transaction their "trust bound". This number might be different for every person.

The more service that is provided, the more trust either the client is putting on the provider or the provider is putting on the client. Therefore, in a decentralized system, where clients and providers may not trust each other, it becomes advantageous to make this amount of service as small as possible. However, each transaction has an associated transaction fee, making it advantageous to have as few transactions as possible to keep this cost down. These two facts seem to be completely at odds with one another, but this is where probabilistic nanopayments come in. Instead of actually sending a small amount of value, the client sends what we call a "lottery ticket" with an expected value of the intended payment. These only pay out when the recipient receives a winning ticket

3 The Law of Large Numbers

Imagine you are flipping a (fair, two-sided) coin repeatedly and 100 times in a row, the coin lands heads up. This is already unexpected. In fact, if you flip a (fair, two-sided) coin 100 times in a row, there is a 1 in $1,267,650,600,228,229,401,496,703,205,376$ chance that the coin lands heads up every single time. So what about the next coin flip? If you flip a coin 101 times in a row, there is an even smaller chance that the coin lands heads up every time- a 1 in $2,535,301,200,456,458,802,993,406,410,752$ chance. It certainly *feels* like there is a higher chance of getting a tails on the 101th coin flip, since the last hundred were heads. This is what we call the gambler's fallacy-the idea that previous outcomes in an (independent and identically distributed) experiment can effect the outcome of the next round of the experiment. And it is a fallacy for a reason. Every time you flip a (fair, two-sided) coin, there is a fifty percent chance of receiving a heads, regardless of what has happened before.

But there is a kernel of an idea there. The gambler's fallacy feels like it should be true because if you flip a coin a whole bunch of times, it seems like you should get heads roughly fifty percent of the time. That statement actually *is* true. That is, **if you flip a (fair, two-sided) coin a *large number* of times, the coin will land heads up close to fifty percent of the time**.

I have to say though, as a mathematician, this statement drives me crazy. In my field, we like rigorous statements that have unambiguous meaning, so the phrases "large number" and "close to" just feel wrong. However, it turns out that there is a way to "rigorize" this statement[4]. And, as mathematicians tend to do, we'll make this statement more general as well.

For $i = 1, 2, \ldots$, let r_i be an infinite sequence of (independent, identically distributed, Lebesgue integrable) random variables with expected value $E(r_1) = E(r_2) = ... = \mu$. Then,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} r_i = \mu$$

Here, for any n, $\frac{1}{n} \sum_{i=1}^{n} r_i$ is the average. In words: **if a probability experiment is repeated a large number of times, the average outcome will converge to the expected value**.

4 Nanopayments

A probabalistic Nanopayment has a face value, that is, the value that the Nanopayment is worth if it is a winner, and we call this value V. It also has a probability of winning, which we call P. Then $P \cdot V$ is the expected value of the Nanopayment. To decide if the Nanopayment is a winner (meaning that its true value is the face value), the Sender and Receiver of the Nanopayment need to choose a random number *together*. I like to think of this as if they were in the same location, they might pull a marble out of a bag, or spin one of those prize wheels. But in most cases, the Sender and Receiver are complete strangers, and might be on opposite sides of the world. So how do they choose a number in such a way that each party can trust that the other was not able to influence the outcome to their advantage? The answer uses the magic of *hash functions*.

Definition 4.1. A *cryptographic hash function* is a map from an arbitrary binary string to a binary string of some fixed length such that:

1. The function is one way. In other words, for any specified output, it is not computationally feasible to find an input that maps to it.

2. The function is collision-resistant. In other words, it is computationally infeasible to find two distinct inputs that have the same output.

The output of a cryptographic hash function is often referred to simply as a "hash", and the function applied to a specific input is often referred to as the hash of that input. An example of a cryptographic hash function is SHA-256 (SHA stands for "Secure Hash Algorithm"), which outputs a 256 bit hash. We also occasionally refer to "hashing numbers together". This is done by first combining the numbers in some way, such as adding them, appending them, or multiplying them, and then taking the hash of the combination.

When generating a Nanopayment, the Receiver *commits* to a random[5] number using a cryptographic commitment scheme. This allows the Receiver to keep thier number hidden without being able to change it. I like to visualize this as if the Receiver writes down a number on a slip of paper, puts it inside of a locked box, and gives the box to the Sender but keeps the key until later.

The Sender then sends back a ticket with an attached ticket number[6] as well as the commitment, to the Receiver. The Receiver can later hash together the ticket number with their random number (which the Sender can verify was the one the Receiver committed to), to see if the ticket is a winner. Which hashes are winners is determined by the protocol, but the winners could be something like those whose hashes end in three consecutive zeroes.

[4] "Rigorize" is apparently actually a real word. My math major college friends and I absolutely thought that we made it up though.

[5] It doesn't actually matter if the number was truly chosen randomly since it will be kept hidden and later hashed with a number chosen by the Sender

[6] In practice, this number could be generated from other information, such as the Sender's address, a timestamp, etc.

4.1 Nanopayments and the Law of Large Numbers

A question I get asked a lot is why should people treat Nanopayments like their value is their expected value? If I run a business, I probably want to be paid in a way that is predictable, and these "lottery tickets" don't seem at first glance like something that is predictable. My answer is this: Nanopayments are designed for sending small payments repeatedly. For example, paying for bandwidth by the packet, or streaming by the minute, or generative AI by the prompt. In any of these circumstances, users are sending or receiving a *large number* of payments. And so the Law of Large Numbers applies, which says that the average value of the payments will be close to the expected value. And since the payments are small, the difference is usually minimal. While this potential for difference might seem like a drawback, I truly believe that it is vastly outweighed by the multitude of possibilities that being able to stream payments allows. I am just so tired of having subscription after subscription, overpaying for services that I need but don't use often, and, to be honest, losing even more money when I forget to cancel a subscription.

References

[1] Jake S. Cannell, Justin Sheek, Jay Freeman, Greg Hazel, Jennifer Rodriguez-Mueller, Eric Hou, Brian J. Fox, and Dr. Steven Waterhouse. Orchid: A Decentralized Network Routing Market. `https://www.orchid.com/whitepaper/english.pdf`.

[2] Rafael Pass and Abhi Shelat. Micropayments for decentralized currencies. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 207–218, 2015.