The math of Nxt forging

mthcl[*]

March 14, 2014. Version 0.2.1

Abstract

We discuss the forging algorithm of Nxt from the probabilistic point of view, and obtain explicit formulas and estimates for several important quantities, such as the probability that an account generates a block, the length of the longest sequence of consecutive blocks generated by one account, and the probability that one concurrent blockchain wins over another one.

1 Forging algorithm

In this article we concentrate on the 1-block-per-minute regime, which is not implemented yet. Assume that there are N forging accounts at a given (discrete) time, $B_1, \dots, B_N$ are the corresponding effective balances, and we denote by

$$b_k = \frac{B_k}{B_1 + \cdots + B_N}, \quad k = 1, \dots, N$$

the proportion of total forging power that the kth account has. Then, to determine which account will generate the next block, we take i.i.d. random variables with Uniform distribution on interval $(0,1)$, and the account which maximizes b_k/U_k generates the block; i.e., the label k_0 of the generating account is determined by

$$k_0 = \arg\max_{j\in\{1,\dots,N\}} \frac{b_j}{U_j}. \tag{1}$$

[*]NXT: 5978778981551971141; author's contact information: `e.monetki@gmail.com`, or send a PM at `bitcointalk.org` or `forums.nxtcrypto.org`

We refer to the quantity b_k/U_k as the *weight* of the kth account, and to b_{k_0}/U_{k_0} as the weight of the block. This procedure is called the *main algorithm* (because it is actually implemented in Nxt at this time), or the *U-algorithm* (because the inverse weights have Uniform distribution).

As a general rule, it is assumed that the probability that an account generates a block is proportional to the effective balance, but, in fact, this is only approximately true (as we shall see in Section 2). For comparison, we consider also the following rule of choosing the generating account: instead of (1), we use

$$k_0 = \arg\max_{j\in\{1,\dots,N\}} \frac{b_j}{|\ln U_j|}. \tag{2}$$

The corresponding algorithm is referred to as *Exp-algorithm* (since the inverse weights now have Exponential probability distribution).

2 Probability of block generation

Observe that (see e.g. Example 2a of Section 10.2.1 of [2]) the random variable $|\ln U_j|/b_j$ has Exponential distribution with rate b_j. Since, obviously, for the Exp-algorithm we can rewrite (2) as

$$k_0 = \arg\min_{j\in\{1,\dots,N\}} \frac{|\ln U_j|}{b_j},$$

the inverse weight of the generated block is also an Exponential random variable with rate $b_1 + \cdots + b_N = 1$ (cf. (5.6) of [3]), and the probability that the kth account generates the block is exactly b_k (this follows e.g. from (5.5) of [3]).

However, for U-algorithm the calculation in the general case is not so easy. We concentrate on the following situation, which seems to be critical for accessing the security of the system: N is large, the accounts $2, \dots, N$ belong to "poor honest guys" (so $b_2, \dots, b_N$ are small), and the account 1 belongs to a "bad guy", who is not necessarily poor (i.e., $b := b_1$ need not be very small).

We first calculate the probability distribution of the biggest weight among the good guys: for $x \gg \max_{k\geq 2} b_k$ let us write

$$\begin{aligned}
\mathbb{P}\Big[\max_{k\geq 2}\frac{b_k}{U_k} < x\Big] &= \prod_{k\geq 2}\mathbb{P}\Big[U_k > \frac{b_k}{x}\Big] \\
&= \prod_{k\geq 2}\Big(1-\frac{b_k}{x}\Big) \\
&= \exp\sum_{k\geq 2}\ln\Big(1-\frac{b_k}{x}\Big) \\
&\approx e^{-\frac{1-b}{x}}, \qquad (3)
\end{aligned}$$

since $\ln(1-y) \sim -y$ as $y \to 0$ and $b_2+\ldots+b_N = 1-b$. We calculate now the probability $f(b)$ that the bad guy generates the block, in the following way. Let Y be a random variable with distribution (3) and independent of U_1, and we write (conditioning on U_1)

$$\begin{aligned}
f(b) &:= \mathbb{P}\Big[\frac{b}{U_1} > Y\Big] \\
&= \int_0^1 \mathbb{P}\Big[Y < \frac{b}{z}\Big]\,dz \\
&= \int_0^1 e^{-\frac{1-b}{b}z}\,dz \\
&= \frac{b}{1-b}\Big(1-e^{-\frac{1-b}{b}}\Big). \qquad (4)
\end{aligned}$$

It is elementary to show that $f(b) > b$ for all $b \in (0,1)$ (see also Figure 1), and (using the Taylor expansion) $f(b) = b + b^2 + O(b^3)$ as $b \to 0$.

Conclusions:

- If an account has proportion b of the total effective balance and the forging powers of other accounts are relatively small, then the probability that it generates the next block is given by (4).
- With small b, this is approximately $b + b^2$, i.e., the block generating probability is roughly proportional to the effective balance with a quadratic correction.
- It is also straightforward to obtain that the probability that a good guy k generates a block is $b_k(1-f(b))$, up to terms of smaller order.

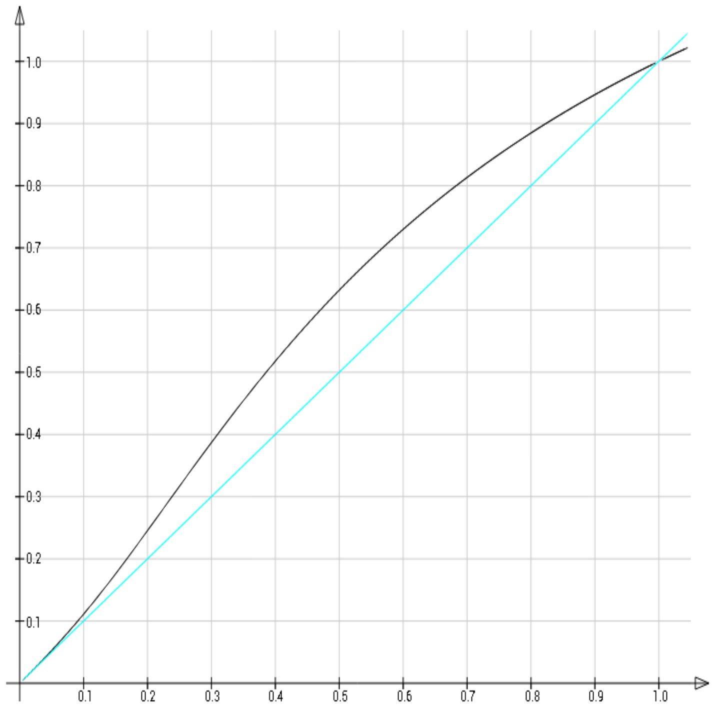

Figure 1: The plot of $f(b)$ (black curve)

3 Longest run

We consider a “static” situation here: there are no transactions (so that the effective balances are equal to full balances and do not change over time). The goal is to be able to find out, how many blocks in a row can be typically generated by a given account over a long period n of time.

So, assume that the probability that an account generates the next block is p (see in Section 2 an explanation about how p can be calculated). It is enough to consider the following question: let R_n be the maximal number of consecutive 1's in the sequence of n Bernoulli trials with success probability p; what can be said about the properties of the random variable R_n?

The probability distribution of R_n has no tractable closed form, but is nevertheless quite well studied, see e.g. [4] (this article is freely available in the internet). The following results are taken from [1]: we have

$$\mathbb{E}R_n = \log_{1/p} qn + \frac{\gamma}{\ln 1/p} - \frac{1}{2} + r_1(n) + \varepsilon_1(n), \tag{5}$$

$$\text{Var}R_n = \frac{\pi^2}{6\ln^2 1/p} + \frac{1}{12} + r_2(n) + \varepsilon_2(n), \tag{6}$$

where $q = 1 - p$, $\gamma \approx 0.577\ldots$ is the Euler-Mascheroni constant, $\varepsilon_{1,2}(n) \to 0$ as $n \to \infty$, and $r_{1,2}(n)$ are uniformly bounded in n and very small (so, in practice, $r_{1,2}$ and $\varepsilon_{1,2}$ can be neglected).

In the same work, one can also find results on the distribution itself. Let W_p be a random variable with Gumbel-type distribution: for $y \in \mathbb{R}$

$$\mathbb{P}[W_p \leq y] = \exp(-p^{y+1}).$$

Then, for $x = 0, 1, 2, \ldots$ it holds that

$$\mathbb{P}[R_n = x] \approx \mathbb{P}[x - \log_{1/p} qn < W_p \leq x + 1 - \log_{1/p} qn], \tag{7}$$

with the error decreasing to 0 as $n \to \infty$. So, in particular, one can obtain that

$$\begin{aligned}\mathbb{P}[R_n \geq x] &\approx 1 - \exp(-p^{x+1}qn)\\ &\approx p^{x+1}qn \end{aligned} \tag{8}$$

if $p^{x+1}qn$ is small (the last approximation follows from the Taylor expansion for the exponent).

For example, consider the situation when one account has 10% of all forging power, and the others are relatively small. Then, according to (4), the probability that this account generates a block is $p \approx 0.111125$. Take $n = 1000000$, then, according to (5)–(7), we have

$$\begin{aligned} \mathbb{E}R_n &\approx 6.00273, \\ \mathrm{Var}R_n &\approx 0.424, \\ \mathbb{P}[R_n \geq 7] &\approx 0.009\,. \end{aligned}$$

Conclusions:

- The distribution of the longest run of blocks generated by one particular account (or group of accounts) is easily accessible, even though there is no exact closed form. Its expectation and variance are given by (5)–(6), and the one-sided estimates are available using (8).

4 Weight of the blockchain and concurrent blockchains

First, let us look at the distribution of the inverse weight of a block. In the case of Exp-algorithm, everything is simple: as observed in Section 2, it has the Exponential distribution with rate 1. This readily implies that the expectation of the sum of inverse weights of n blocks equals n.

As for the U-algorithm, we begin by considering the situation when all relative balances are small. Analogously to (3), being W the weight of the block, for $x \ll (\max_k b_k)^{-1}$ we calculate

$$\begin{aligned} \mathbb{P}\Big[\frac{1}{W} > x\Big] &= \mathbb{P}\Big[\max_k \frac{b_k}{U_k} < \frac{1}{x}\Big] \\ &= \prod_k \mathbb{P}\Big[U_k > xb_k\Big] \\ &= \prod_k (1 - xb_k) \\ &= \exp \sum_{k \geq 2} \ln(1 - xb_k) \\ &\approx e^{-x}, \end{aligned} \tag{9}$$

so also in this case the distribution of the inverse weight is approximately Exponential with rate 1.

We consider now the situation when all balances except the first one are small, and $b := b_1$ need not be small. For the case of U-algorithm, similarly to the above we obtain for $x \in (0, 1/b)$

$$\mathbb{P}\Big[\frac{1}{W} > x\Big] \approx (1 - bx)e^{-(1-b)x}, \tag{10}$$

so

$$\begin{aligned} \mathbb{E}\frac{1}{W} &\approx \int_0^{1/b} (1 - bx)e^{-(1-b)x}\, dx \\ &= \frac{be^{-\frac{1-b}{b}} + 1 - 2b}{(1-b)^2}. \end{aligned} \tag{11}$$

One can observe (see Figure 2) that the right-hand side of (11) is strictly between 1/2 and 1 for $b \in (0, 1)$.

Let us consider now the following attack scenario: account 1 (the "bad guy", with balance b) temporarily disconnects from the network and forges its own blockchain; he then reconnects hoping that his blockchain would be "better" (i.e., has smaller sum of inverse weights). Then, while the account 1 is disconnected, the "good" part of the network produces blocks with inverse weights having Exponential distribution with rate $1-b$, and thus each inverse weight has expected value $\frac{1}{1-b}$.

Let $X_1, X_2, X_3, \ldots$ be the inverse weights of the blocks produced by the "good part" of the network (after the bad guy disconnects), and we denote by $Y_1, Y_2, Y_3, \ldots$ the inverse weights of the blocks produced by the bad guy. We are interested in controlling the probability of the following event (which means that the blockchain produced by the bad guy is better)

$$H_m = \{X_1 + \cdots + X_m - Y_1 - \cdots - Y_m \geq 0\}$$

for "reasonably large" m (e.g., $m = 10$ or so). If the probability of H_m is small, this means that the bad guy just does not have enough power to attack the network; on the other hand, if this probability is not small, then the system should be able to fence off the attack by other means, which we shall not discuss in this note.

We obtain an upper bound on the probability of the event H_m using the so-called Chernoff theorem (see e.g. Proposition 5.2 of Chapter 8 of [2]): we

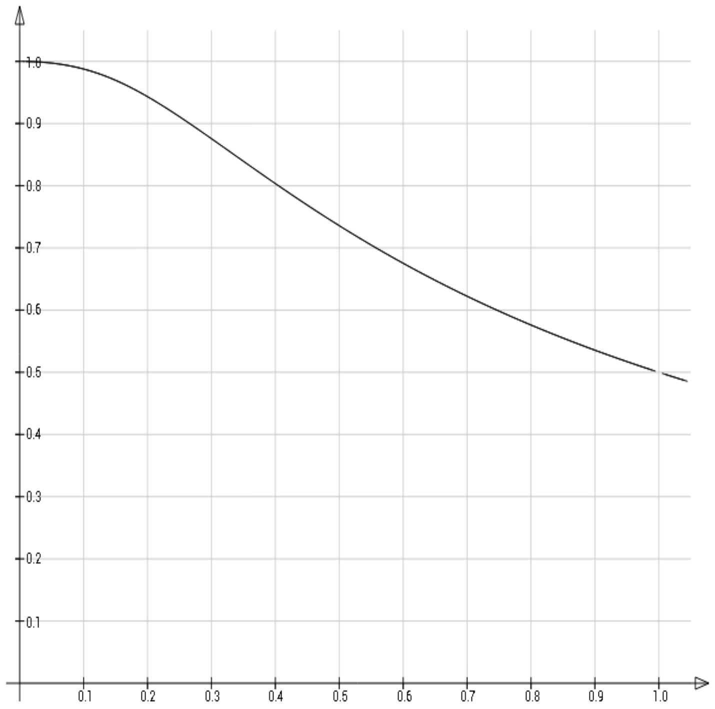

Figure 2: Expectation of the inverse weight (as a function of b)

have

$$\mathbb{P}[H_m] \le \delta^m,$$

where

$$\delta = \inf_{t>0} \mathbb{E}e^{t(X_1-Y_1)}. \tag{12}$$

It is important to observe that this bound is nontrivial (i.e., $\delta < 1$) only in the case $\mathbb{E}X_1 < \mathbb{E}Y_1$.

For U-algorithm, X_1 is Exponentially distributed with rate $1-b$, and Y_1 has Uniform$(0, b^{-1})$ distribution. So, the condition $\mathbb{E}X_1 < \mathbb{E}Y_1$ is equivalent to $(1-b)^{-1} < (2b)^{-1}$, that is, $b < 1/3$. Then, for $b < 1/3$, the parameter δ from (12) is determined by

$$\delta = \delta(b) = b(1-b) \inf_{0<t<1-b} \frac{1-e^{-t/b}}{t(1-b-t)} \tag{13}$$

(see the plot of $\delta(b)$ on Figure 3), so we have

$$\mathbb{P}[H_m] \le \delta(b)^m. \tag{14}$$

For example, for $b = 0.1$ we have $\delta(b) \approx 0.439$. We have, however, $\delta(b) \approx 0.991$ for $b = 0.3$, which means that means that one has to take very large m in order to make the right-hand side of (14) small in this case.

For the Exp-algorithm, the bad guy would produce blocks with inverse weights having Exponential distribution with rate b, so each inverse weight has expected value $\frac{1}{b}$. Similarly to the above, one obtains that the condition $\mathbb{E}X_1 < \mathbb{E}Y_1$ is equivalent to $b < 1/2$, and

$$\mathbb{P}[H_m] \le (4b(1-b))^m \tag{15}$$

(that is, δ can be explicitly calculated in this case and equals $4b(1-b)$; observe that $4b(1-b) < 1$ for $b < 1/2$).

Conclusions:

- We analyse an attack strategy when one account (or a group of accounts) temporarily disconnects from the main network and tries to forge a "better" blockchain than the one forged by other accounts, in the situation when one bad rich guy has proportion b of total amount of NXT, and the stakes of the others are relatively small.

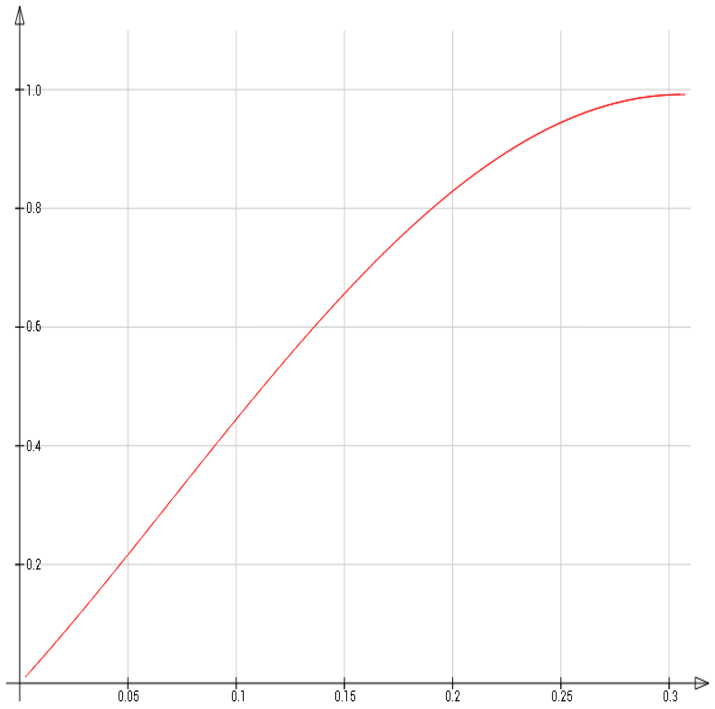

Figure 3: The plot of $\delta(b)$

- The probability that the bad guy forges a better chain of length m can be controlled using (14) (for the U-algorithm) or (15) (for the Exp-algorithm).
- It should be observed that this probability does not tend to 0 (as $m \to \infty$) if the bad guy has at least 1/3 of all *active* balances in the network in the case of U-algorithm (correspondingly, at least 1/2 in the case of Exp-algorithm). There should exist some specific methods for protecting the network against such an attack in the case when there is risk that (active) relative balance of the bad guy could become larger than the above threshold.
- For the current realization of the U-algorithm, the author expects that this analysis can be performed in a quite similar way (because the inverse weight is then proportional to the time to the next block, and the longest blockchain wins), with an additional difficulty due to the oscillating `BaseTarget`.
- It may be a good idea to limit the forging power of accounts by some fixed threshold, e.g., if an account has more than, say, 300K NXT, then it forges as if it had exactly 300K NXT. Of course, a rich guy can split his fortune between smaller accounts, but then all those accounts would forge roughly as one big account (without threshold) under Exp-algorithm. So, one can use the computationally easier U-algorithm without having the drawbacks (the 1/3 vs. 1/2 issue) discussed in this section.

References

[1] L. Gordon, M.F. Schilling, M.S. Waterman (1986) An extreme value theory for long head runs. *Probab. Theory Relat. Fields* **72**, 279–287.

[2] Sheldon M. Ross (2009) *A First Course in Probability.* 8th ed.

[3] Sheldon M. Ross (2012) *Introduction to Probability Models.* 10th ed.

[4] Mark Schilling (1990) The Longest Run of Heads. *The College Math J.*, **21** (3), 196–206.