On the timestamps in the tangle

Serguei Popov*

August 6, 2017

Description of the problem and proposed algorithms

In the following we consider the tangle [1] at a fixed moment of time; that is, the state of the tangle is fixed, and so we denote it simply by $\mathbb{T}$. Some notations: if x is a site (transaction) on the tangle, we denote by $\mathcal{A}(x)$ the set of the two transactions approved by x. We say that x *references* (indirectly approves) y if there is a sequence of sites $x = x_0, x_1, \ldots, x_k = y$ such that $x_j \in \mathcal{A}(x_{j-1})$ for all $j = 1, \ldots, k$. Let us write

$$\mathcal{P}(x) = \big\{y \in \mathbb{T} : y \text{ is referenced by } x\big\},$$
$$\mathcal{F}(x) = \big\{z \in \mathbb{T} : z \text{ references } x\big\}$$

for the "past" and the "future" with respect to x. In other words, the above introduces a *partial order* structure on the tangle. Also we denote by $\mathrm{Ind}(x) = \mathbb{T} \setminus \big(\mathcal{P}(x) \cup \mathcal{F}(x)\big)$ the set of transactions which neither reference nor are referenced by x (i.e., which are, in a way, *independent* from x, hence the abbreviation used). Observe that, by definition, $x \in \mathrm{Ind}(x)$.

Next, we assume that for each transaction x there is a *timestamp* $\mathbf{t}(x)$; that is, in principle, $\mathbf{t}(x)$ corresponds to the time when the transaction was attached to the tangle. It is required that $\mathbf{t}(x) > \mathbf{t}(y_{1,2})$ where $\mathcal{A}(x) = \{y_1, y_2\}$, for all $x \in \mathbb{T}$; in other words, a transaction cannot approve another transaction whose timestamp is "from the future". Our standing assumption will be that the *large majority* (in *some* reasonable sense) of the transactions were issued by honest nodes equipped with (more-or-less) reliable clocks.

*a.k.a. `mthcl`; author's contact information: `serguei.popov@iota.org`

Now, consider some transaction x (which is, typically, already "deep inside" the tangle), it has the timestamp $\mathbf{t}(x)$. The problem is that we do not know if it was issued by a honest or malicious node (or, maybe, it could be issued by a honest node whose clock is wrong for some reason), therefore we cannot be sure if $\mathbf{t}(x)$ is (even approximately) the real time T_x when x was attached to the tangle. Our goal is to construct the *confidence interval* $[a_x, b_x]$ for T_x; that is, we want (deterministic) $a_x \leq b_x$ such that the event $\{T_x \in [a_x, b_x]\}$ occurs with high probability.

In the following, we consider two procedures for constructing such an interval.

Procedure 1. Fix $\beta \in (0, \frac{1}{2})$, and consider the data collection (in fact, a multiset)

$$\big(t(y) : y \in \mathrm{Ind}(x)\big). \tag{1}$$

Then, define a_x and b_x to be the β- and $(1-\beta)$-quantiles of the above data collection, correspondingly.

We have to explain why it is not a good idea to have $\beta = 0$ (i.e., take a_x and b_x the minimal and the maximal values in the above data collection). The reason is that a malicious entity can mess with the procedure, by pushing a_x to 0 (by issuing new transactions that approve other transactions that are deep in the past, and have small $\mathbf{t}$-values) and b_x to infitity (by issuing new transactions that do not reference x and have very large $\mathbf{t}$-values). Those "disrupting" transactions will be cut off if β is sufficiently away from 0.

In principle, by increasing β we also increase our defences against the malicious behaviour described above; on the other hand, if β is "too close" to $\frac{1}{2}$, the result would be "too random" (observe that, for β close to $\frac{1}{2}$, $\mathbf{t}(x)$ itself may not make it to the confidence interval, even in the case when x was issued by a honest node!). It is unclear, for now, what would be the "optimal" value of β; in fact, one has to make assumptions on the proportion of the malicious nodes in the network and their modus operandi to perform such an analysis. In any case, as a rule of thumb, a value of β in $[0.2, 0.3]$ would probably work.

As a drawback of the above procedure, note that one has to do the calculations for each x separately (observe that, in general, $\mathrm{Ind}(x) \neq \mathrm{Ind}(y)$ when $x \neq y$); this may pose computational difficulties in case when the tangle is very large. On the other hand, the outcome of the procedure is deterministic, provided of course that the nodes use the same β and see the same state of the tangle.

Let us now describe another procedure, which is computationally easier and works for many transactions at once; on the other hand, it produces a *random*[1] result.

[1]that is, different nodes may arrive to different confidence intervals for T_x even if they see the

Procedure 2. Run the random walk described in Section 4.1 of [1] starting from some site deep inside the tangle, and take a_x to be the timestamp of the *last* transaction referenced by x, and b_x to be the timestamp of the *first* transaction that references x.

Observe that, if a malicious node tries to forge the timestamp of a fixed transaction, this transaction is unlikely to be picked by the random walk. Indeed, in case the malicious node puts the timestamp "from the past" (i.e., $\mathbf{t}(x)$ is much less than the real time T_x when x was issued), this corresponds to the "lazy tip" case of Section 4.1 of [1]; if the malicious node puts the timestamp "from the future" (i.e., $\mathbf{t}(x)$ is much greater than the real time when x was issued), such a transaction would not be referenced by anyone for long time, which again makes it unlikely that the random walk eventually passes through it.

Conclusion

The tangle is a graph with only a partial order structure, which makes it difficult (in fact, generally impossible) to establish the correct *time order* of transactions. Even if all transactions have timestamps on them, we cannot be sure that all these timestamps are accurate (there can be some malicious nodes that want to fool the network about the true time when their transactions appear, and/or some nodes with a wrong clock). Nevertheless, one can determine the *confidence intervals* for timestamps with reasonable accuracy. In the above text we described two possible algorithms for doing that; the first one is computationally more difficult, but produces a deterministic result (two nodes that use the same β and see the same state of the tangle will get the same confidence interval). The other algorithm is simpler and works for determining the timestamps' confidence intervals for several transactions at once, but produces a random result (it depends on the path that the random walk have actually chosen).

References

[1] S. Popov (2015) The tangle. `https://iota.org/IOTA_Whitepaper.pdf`

same tangle