TxFlow Functional Form

NearProtocol

Last Updated November 27, 2018

TxFlow algorithm is all about building a DAG of messages and recognizing when a certain message has some special properties. The properties of each message depend exclusively on its sub-DAG. We constantly iterate on the improvements of the algorithm in several languages that use different algorithmic approaches, e.g. Rust implementation tries to avoid DAG traversals by precomputing the needed information in each message, while Javascript aims for the simplicity of the implementation. While we do these iterations it easy to lose track of the actual algorithm in the implementation details. This notebook lists the predicates that formally define the properties of the messages and serves as the **source of truth** for TxFlow.

Basic notation

We say that message c directly approves p if p contains the hash of c. We say that c approves p, denoted as $p \leftarrow c$ if there is p' such that c directly approves p' and p' approves p, i.e. it is a transitive closure. We also allow it to be reflexive so $c \leftarrow c$ is always true.

The domain of a message c is all messages that it approves:

$$\overline{D}(c) \triangleq \{p | p \leftarrow c\}$$
$$D(c) \triangleq \overline{D}(c) \setminus \{c\}$$

When it matters we annotate the message with the witness that created it and its epoch:

$$m_w^e \text{ – message with epoch } e\text{, created by } w;$$

$$L(e) \text{ – a leader of epoch } e;$$

$$S(c, p) \text{ – message } c \text{ contains signature for message } p\text{, where } p \leftarrow c;$$

Definitions

Representative message

$$\begin{aligned}
R(m_w^x, y) \triangleq & \quad \textit{Message } m_w^x \textit{ is a representative of epoch } y \textit{ if:} \\
[\nexists m' \in D(m_w^x), y' : y' \geq y \wedge R(m', y')] \wedge & \quad \textit{This message does not approve representative with epoch} \geq y. \\
L(y) = w \wedge & \quad \textit{It is the leader of epoch } y. \\
[[x = y \wedge \{x = 0 \vee \exists r \in D(m_w^x) : R(r, y-1)\}] \vee & \quad \textit{It is the same epoch and it approves the prev representative if it is not zero epoch} \\
x > y \wedge \exists k_w^{y-1} \in D(m_w^x) : SP(m_w^x, k)] & \quad \textit{...or it witnesses that its own kickout has enough promises.}
\end{aligned}$$

Kickout message

$$\begin{aligned}
K(m_w^x, x-1) \triangleq & \quad \textit{Message } m_w^x \textit{ kicks out the representative of epoch } x-1 \textit{ if:} \\
[\nexists m' \in D(m_w^x), y' : y' \geq x \wedge R(m', y')] \wedge & \quad \textit{This message does not approve representative with epoch} \geq x. \\
L(x) = w \wedge & \quad \textit{It is the leader of epoch } x. \\
x > 0 \wedge & \quad \textit{There is a previous epoch.} \\
\nexists m' \in D(m_w^x) : R(m', x-1) & \quad \textit{There is no representative that we are kicking out.}
\end{aligned}$$

Endorsable

$$\begin{aligned}
\widetilde{E}(m_w, n) \triangleq & \quad \textit{Message } n \textit{ is endosable by } m_w \textit{ if:} \\
n \in \overline{D}(m_w) \wedge & \quad \textit{It is approved by it. We allow representative to endorse itself.} \\
\exists e : R(n, e) \wedge & \quad n \textit{ is actually a representative of some epoch } e. \\
\nexists m'_w \in D(m_w) : \widetilde{E}(m'_w, n) \wedge & \quad \textit{We did not already endorse it with some other message.} \\
\nexists e, p_w, k \in D(m_w) : K(k, e) \wedge R(m_w, e) \wedge P(p_w, k) & \quad \textit{The same witness did not already promise to kickout the very same epoch.}
\end{aligned}$$

Endorsement message

$$E(m_w, n) \triangleq \widetilde{E}(m_w, n) \wedge S(m_w, n) \quad \textit{The message } m_w \textit{ endorses } n \textit{ if } n \textit{ is endorsable by } m_w \textit{ and is signed.}$$

We separate the definition of the endorsement into two: **Endorsable** and **Endorsement** to workaround the cases when a malicious witness should have signed a representative but did not. In that case, the message should not count towards the representative message, but it carries the same constraints for the witness, e.g. they cannot promise to kickout that message later.

Promise message

$$\begin{aligned}
P(p_w, k) \triangleq & & \textit{Message } p_w \textit{ promises to support kickout } k \textit{ if:} \\
p_w \in \overline{D}(p_w) \wedge & & \textit{It is approved by it. We allow kickout message to promise to itself.} \\
\exists e : K(k, e) \wedge & & \textit{It is actually a kickout of some epoch } e\textit{.} \\
\nexists p'_w \in D(p_w) : P(p'_w, k) \wedge & & \textit{We did not already give a promise with some other message.} \\
\nexists e, m_w, r \in \overline{D}(p_w) : R(m_w, e) \wedge K(k, e) \wedge \widetilde{E}(m_w, r) & & \textit{The same witness did not already produce and endorsement(*) for the very same epoch.}
\end{aligned}$$

Notice the differences in the last line of the **Endorsable** and **Promise message**:

- First, if the message simultaneously approves both kickout and a representative than it is a endorsement, and not a kickout (see the D vs $\overline{D}$ asymetry that breaks the tie);
- (*) Second, notice how we use $\widetilde{E}$ instead of E in the **Promise message**. Even if we previously did not endorse the representative but we should have (we did not provide the signature) than we are still not allowed to promise to a kickout for that epoch.

Sufficiently endorsed

The message m has a proof that a representative n has endorsements from more than $2/3$ of the witnesses.

$$SE(m, n) \triangleq |\{w | \exists m'_w \in \overline{D}(m) : E(m'_w, n)\}| > 2N/3$$

Sufficiently promised

The message p has a proof that a kickout k has promises from more than $2/3$ of the witnesses.

$$SP(p, k) \triangleq |\{w | \exists p'_w \in \overline{D}(p) : E(p'_w, k)\}| > 2N/3$$