1 Payment Network Functionality

Functionality $\mathcal{F}_{\text{PayNet}}$ – interface

– from $\mathcal{E}$:
- (REGISTER, delay, relayDelay)
- (TOPPEDUP)
- (OPENCHANNEL, *Alice*, *Bob*, x, *tid*)
- (CHECKFORNEW, *Alice*, *Bob*, *tid*)
- (PAY, *Bob*, x, $\overrightarrow{\texttt{path}}$, `receipt`)
- (CLOSECHANNEL, `receipt`, *pchid*)
- (FORCECLOSECHANNEL, `receipt`, *pchid*)
- (POLL) - obsolete
- (PUSHFULFILL, *pchid*) - obsolete
- (PUSHADD, *pchid*) - obsolete
- (COMMIT, *pchid*) - obsolete
- (FULFILLONCHAIN) - obsolete
- (GETNEWS)

– to $\mathcal{E}$:
- (REGISTER, *Alice*, `delay`(*Alice*), `relayDelay`(*Alice*), pubKey)
- (REGISTERED)
- (NEWS, newChannels, closedChannels, updatesToReport)

– from $\mathcal{S}$:
- (REGISTERDONE, *Alice*, pubKey)
- (CHANNELANNOUNCED, *Alice*, $p_{Alice,F}$, $p_{Bob,F}$, *fchid*, *pchid*, *tid*)
- (UPDATE, `receipt`, *Alice*) - obsolete
- (CLOSEDCHANNEL, `channel`, *Alice*)
- (RESOLVEPAYS, *payid*, `charged`) - obsolete

– to $\mathcal{S}$:
- (REGISTER, *Alice*, delay, relayDelay)
- (OPENCHANNEL, *Alice*, *Bob*, x, *fchid*, *tid*)
- (CHANNELOPENED, *Alice*, *fchid*)
- (PAY, *Alice*, *Bob*, x, $\overrightarrow{\texttt{path}}$, `receipt`, *payid*) - obsolete
- (CONTINUE) - obsolete
- (CLOSECHANNEL, *fchid*, *Alice*)
- (FORCECLOSECHANNEL, *fchid*, *Alice*)
- (POLL, Σ_{Alice}, *Alice*) - obsolete
- (PUSHFULFILL, *pchid*, *Alice*) - obsolete
- (PUSHADD, *pchid*, *Alice*) - obsolete
- (COMMIT, *pchid*, *Alice*) - obsolete
- (FULFILLONCHAIN, t, *Alice*) - obsolete

Fig. 1.

Functionality $\mathcal{F}_{\text{PayNet}}$ – registration and corruption

```
1: Initialisation:
2:     channels, pendingPay, pendingOpen, corrupted, $\Sigma \leftarrow \emptyset$

3: Upon receiving (REGISTER, delay, relayDelay) from $Alice$:
4:     delay ($Alice$) ← delay // Must check chain at least once every
       delay($Alice$) blocks
5:     relayDelay ($Alice$) ← relayDelay
6:     updatesToReport ($Alice$) , newChannels ($Alice$) ← $\emptyset$
7:     polls ($Alice$) ← $\emptyset$
8:     focs ($Alice$) ← $\emptyset$
9:     send (READ) to $\mathcal{G}_{\text{Ledger}}$ as $Alice$, store reply to $\Sigma_{Alice}$, add $\Sigma_{Alice}$ to $\Sigma$ and
       add largest block number to polls($Alice$)
10:    checkClosed($\Sigma_{Alice}$)
11:    send (REGISTER, $Alice$, delay, relayDelay) to $\mathcal{S}$

12: Upon receiving (REGISTERDONE, $Alice$, pubKey) from $\mathcal{S}$:
13:    pubKey ($Alice$) ← pubKey
14:    send (REGISTER, $Alice$, delay($Alice$), relayDelay($Alice$), pubKey) to $Alice$

15: Upon receiving (TOPPEDUP) from $Alice$:
16:    send (READ) to $\mathcal{G}_{\text{Ledger}}$ as $Alice$ and store reply to $\Sigma_{Alice}$
17:    checkClosed($\Sigma_{Alice}$)
18:    assign the sum of all output values that are exclusively spendable by $Alice$
       to onChainBalance
19:    send (REGISTERED) to $Alice$

20: Upon receiving any message ($M$) except for (REGISTER) or (TOPPEDUP) from
    $Alice$:
21:    if if haven't received (REGISTER) and (TOPPEDUP) from $Alice$ (in this
       order) then
22:        send (INVALID, $M$) to $Alice$ and ignore message
23:    end if
```

Fig. 2.

Functionality $\mathcal{F}_{\text{PayNet}}$ – open

```
1: Upon receiving (openChannel, $Alice, Bob, x, tid$) from $Alice$:
2:     ensure $tid$ hasn't been used by $Alice$ for opening another channel before
3:     choose unique channel ID $fchid$
4:     pendingOpen $(fchid) \leftarrow (Alice, Bob, x, tid)$
5:     send (openChannel, $Alice, Bob, x, fchid, tid$) to $\mathcal{S}$

6: Upon receiving (channelAnnounced, $Alice$, $p_{Alice,F}$, $p_{Bob,F}$, $fchid$, $pchid$, $tid$)
   from $\mathcal{S}$:
7:     ensure that there is a pendingOpen($fchid$) entry with temporary id $tid$
8:     add $p_{Alice,F}, p_{Bob,F}$, $pchid$ and mark "$Alice$ announced" to
   pendingOpen($fchid$)

9: Upon receiving (checkForNew, $Alice$, $Bob$, $tid$) from $Alice$:
10:     ensure there is a matching channel in pendingOpen($fchid$), marked with
   "$Alice$ announced"
11:     $(\text{funder}, \text{fundee}, x, p_{Alice,F}, p_{Bob,F}) \leftarrow$ pendingOpen $(fchid)$
12:     send (read) to $\mathcal{G}_{\text{Ledger}}$ as $Alice$ and store reply to $\Sigma_{Alice}$
13:     checkClosed($\Sigma_{Alice}$)
14:     ensure that there is a TX $F \in \Sigma_{Alice}$ with a $(x, (p_{\text{funder},F} \land p_{\text{fundee},F}))$
   output
15:     mark channel with "waiting for fundingLocked"
16:     send (fundingLocked, $Alice$, $\Sigma_{Alice}$, $fchid$) to $\mathcal{S}$

17: Upon receiving (fundingLocked, $fchid$) from $\mathcal{S}$:
18:     ensure a channel is in pendingOpen($fchid$), marked with "waiting for
   fundingLocked" and replace mark with "waiting for channelOpened"
19:     send (read) to $\mathcal{G}_{\text{Ledger}}$ as $Bob$ and store reply to $\Sigma_{Bob}$
20:     checkClosed($\Sigma_{Bob}$)
21:     ensure that there is a TX $F \in \Sigma_{Bob}$ with a $(x, (p_{\text{funder},F} \land p_{\text{fundee},F}))$
   output
22:     add receipt(channel) to newChannels($Bob$)
23:     send (fundingLocked, $Bob$, $\Sigma_{Bob}$, $fchid$) to $\mathcal{S}$

24: Upon receiving (channelOpened, $fchid$) from $\mathcal{S}$:
25:     ensure a channel is in pendingOpen($fchid$), marked with "waiting for
   channelOpened" and remove mark
26:     offChainBalance (funder) $\leftarrow$ offChainBalance (funder) $+ x$
27:     onChainBalance (funder) $\leftarrow$ onChainBalance (funder) $- x$
28:     channel $\leftarrow (\text{funder}, \text{fundee}, x, 0, 0, fchid, pchid)$
29:     add channel to channels
30:     add receipt(channel) to newChannels($Alice$)
31:     clear pendingOpen($fchid$) entry
```

Fig. 3.

Functionality $\mathcal{F}_{\mathrm{PayNet}}$ – pay (updated)

```
1: Upon receiving $\left(\textsc{pay}, Bob, x, \overrightarrow{\texttt{path}}\right)$ from $Alice$:
2:     ensure that $\overrightarrow{\texttt{path}}$ consists of open channels that form a path of capacity at least $x$ (in the right direction) from $Alice$ to $Bob$
3:     in every channel $\in \overrightarrow{\texttt{path}}$, reduce balance of party closer to payer by $x$ and increase balance of party closer to payee by $x$
4:     for every channel $\in \overrightarrow{\texttt{path}}$, add receipt of new balance to both parties' updatesToReport
```

Fig. 4.

Functionality $\mathcal{F}_{\mathrm{PayNet}}$ – close

```
1: Upon receiving (closeChannel, receipt, $pchid$) from $Alice$
2:     ensure that there is a channel $\in$ channels : receipt (channel) = receipt with ID $pchid$
3:     retrieve $fchid$ from channel
4:     add ($fchid$, receipt(channel), $\infty$) to pendingClose($Alice$)
5:     do not serve any other (pay, closeChannel) message from $Alice$ for this channel
6:     send (closeChannel, receipt, $pchid$, $Alice$) to $\mathcal{S}$

7: Upon receiving (forceCloseChannel, receipt, $pchid$) from $Alice$
8:     retrieve $fchid$ from channel
9:     add ($fchid$, receipt(channel), $\perp$) to pendingClose($Alice$)
10:    do not serve any other (pay, closeChannel, forceCloseChannel) message from $Alice$ for this channel
11:    send (forceCloseChannel, receipt, $pchid$, $Alice$) to $\mathcal{S}$

12: Upon receiving (closedChannel, channel, $Alice$) from $\mathcal{S}$:
13:    remove any ($fchid$ of channel, receipt(channel), $\infty$) from pendingClose($Alice$)
14:    add ($fchid$ of channel, receipt(channel), $\perp$) to closedChannels($Alice$) // trust $\mathcal{S}$ here, check on checkClosed()
15:    send (continue) to $\mathcal{S}$
```

Fig. 5.

Functionality $\mathcal{F}_{\mathrm{PayNet}}$ – `checkClosed()`

```
function checkClosed($\Sigma_{Alice}$) // Called after every (READ), ensures requested closes eventually happen
    if there is any closing/commitment transaction in $\Sigma_{Alice}$ with no corresponding entry in pendingClose($Alice$) $\cup$ closedChannels($Alice$) then
        add ($fchid$, receipt, $\perp$) to closedChannels($Alice$), where $fchid$ is the ID of the corresponding channel, receipt comes from the latest channel state
    end if
    for all entries ($fchid$, receipt, $h$) $\in$ pendingClose($Alice$) $\cup$ closedChannels($Alice$) do
        if there is a closing/commitment transaction in $\Sigma_{Alice}$ for open channel with ID $fchid$ with a balance that corresponds to receipt then
            let $x, y$ $Alice$'s and channel counterparty $Bob$'s balances respectively
            $\text{offChainBalance}\,(Alice) \leftarrow \text{offChainBalance}\,(Alice) - x$
            $\text{onChainBalance}\,(Alice) \leftarrow \text{onChainBalance}\,(Alice) + x$
            $\text{offChainBalance}\,(Bob) \leftarrow \text{offChainBalance}\,(Bob) - y$
            $\text{onChainBalance}\,(Bob) \leftarrow \text{onChainBalance}\,(Bob) + y$
            remove channel from channels & entry from pendingClose($Alice$)
            if there is an ($fchid$, _, _) entry in pendingClose($Bob$) then
                remove it from pendingClose($Bob$)
            end if
        else if there is a tx in $\Sigma_{Alice}$ that is not a closing/commitment tx and spends the funding tx of the channel with ID $fchid$ then
            halt // DS forgery
        else if there is a commitment transaction in block of height $h$ in $\Sigma_{Alice}$ for open channel with ID $fchid$ with a balance that does not correspond to the receipt and the delayed output has been spent by the counterparty then
            if polls($Alice$) contains an entry in $[h, h + \texttt{delay}(Alice) - 1]$ then
                halt
            else
                negligent($Alice$) $\leftarrow$ true
            end if
        else if there is no such closing/commitment transaction $\wedge\ h = \perp$ then
            assign largest block number of $\Sigma_{Alice}$ to $h$ of entry
        else if there is no such closing/commitment transaction $\wedge\ h \neq \perp\ \wedge$ (largest block number of $\Sigma_{Alice}$) $\geq h + (2 + r)\,\texttt{windowSize}$ then
            halt
        end if
    end for
    if $Alice$ has no open channels in $\Sigma_{Alice}$ AND negligent($Alice$) = false then
        if offChainBalance($Alice$) $\neq 0$ OR onChainBalance($Alice$) is not equal to the total funds exclusively spendable by $Alice$ in $\Sigma_{Alice}$ then
            halt
        end if
    end if
end function
```

Fig. 6.

Functionality $\mathcal{F}_{\text{PayNet}}$ – get news (updated)

```
1: Upon receiving (GETNEWS) from $Alice$:
2:     clear newChannels($Alice$), closedChannels($Alice$), updatesToReport($Alice$)
       and send them to $Alice$ with message name NEWS, stripping $fchid$ and h from
       closedChannels($Alice$)
```

Fig. 7.