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

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

Upon receiving (REGISTER, delay, relayDelay) from $Alice$:
    delay $(Alice) \leftarrow$ delay // Must check chain at least once every delay($Alice$) blocks
    relayDelay $(Alice) \leftarrow$ relayDelay
    updatesToReport $(Alice)$ , newChannels $(Alice) \leftarrow \emptyset$
    polls $(Alice) \leftarrow \emptyset$
    focs $(Alice) \leftarrow \emptyset$
    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($Alice$)
    checkClosed($\Sigma_{Alice}$)
    send (REGISTER, $Alice$, delay, relayDelay) to $\mathcal{S}$

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

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

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

Fig. 2.

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

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

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

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

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

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

Fig. 3.

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

```
Upon receiving $\left(\text{PAY}, Bob, x, \overrightarrow{\texttt{path}}\right)$ from $Alice$:
    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 $Bob$
    starting on $|\overrightarrow{\texttt{path}}|$ clock ticks after receiving this message, on every clock tick, channel $\in \overrightarrow{\texttt{path}}$, reduce balance of party closer to payer by $x$ and increase balance of party closer to payee by $x$ in the next channel on the $\overrightarrow{\texttt{path}}$ and add receipt of new balance to both parties' updatesToReport, starting from the unique channel in which the payee is participating
```

Fig. 4.

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

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

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

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

Fig. 5.

Functionality $\mathcal{F}_{\text{PayNet}}$ – 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
            offChainBalance ($Alice$) $\leftarrow$ offChainBalance ($Alice$) $- x$
            onChainBalance ($Alice$) $\leftarrow$ onChainBalance ($Alice$) $+ x$
            offChainBalance ($Bob$) $\leftarrow$ offChainBalance ($Bob$) $- y$
            onChainBalance ($Bob$) $\leftarrow$ 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 $\Sigma_{Alice}$) $\geq h + (2 + r)$ 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)

```
Upon receiving (getNews) from $Alice$:
    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.

- The functionality above provides unobservability of off-chain payments and can be realised by a protocol in which all players communicate with every other player on every round – sending garbage to the ones with which they don't have to interact. Such a protocol has n^2 communication cost. Indistinguishability holds only in case of a global passive adversary (no corruptions).
- We can also assert unobservability for paths that consist of honest parties only in the case where there is a system-wide maximum path length l and corruptions activate only after $2l$ clock ticks. Both the functionality and the protocol would be the same.
- In case of a normal corruption model, the functionality has to leak the previous and next player on the path, along with the payment value, to a corrupted player that is to receive its message on this clock tick. Also, the functionality has to wait for confirmation from the corrupted player before sending the message to the next player (but this isn't strictly about privacy).