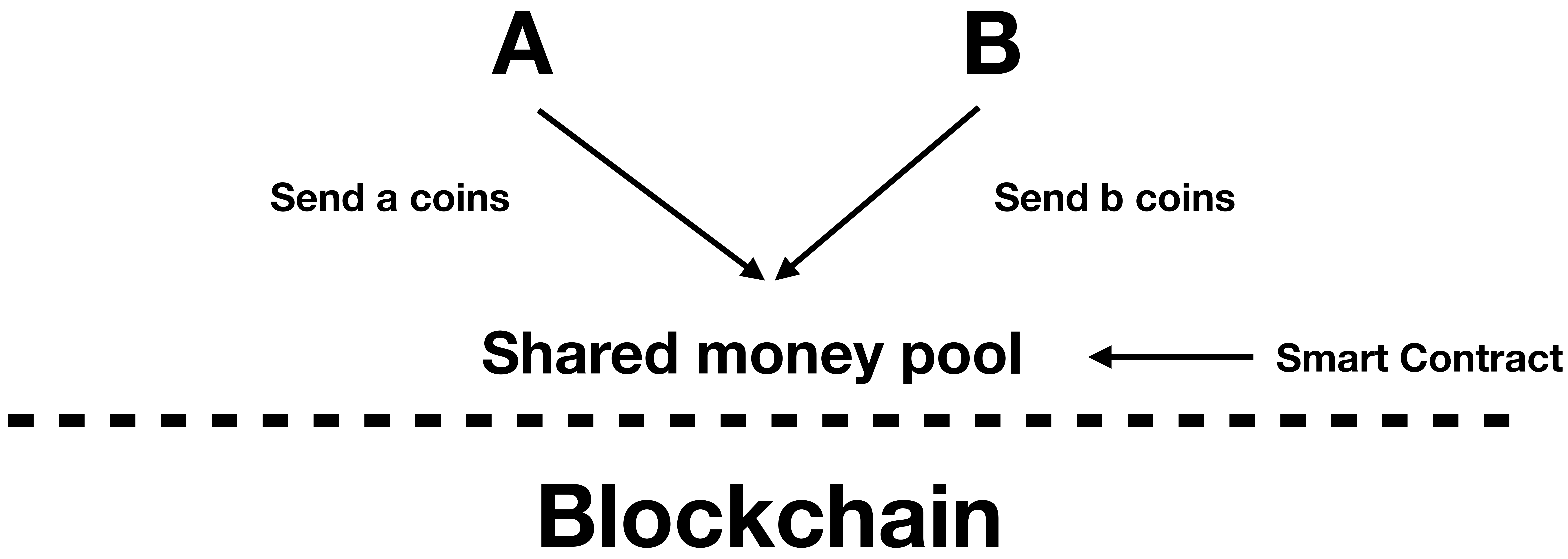
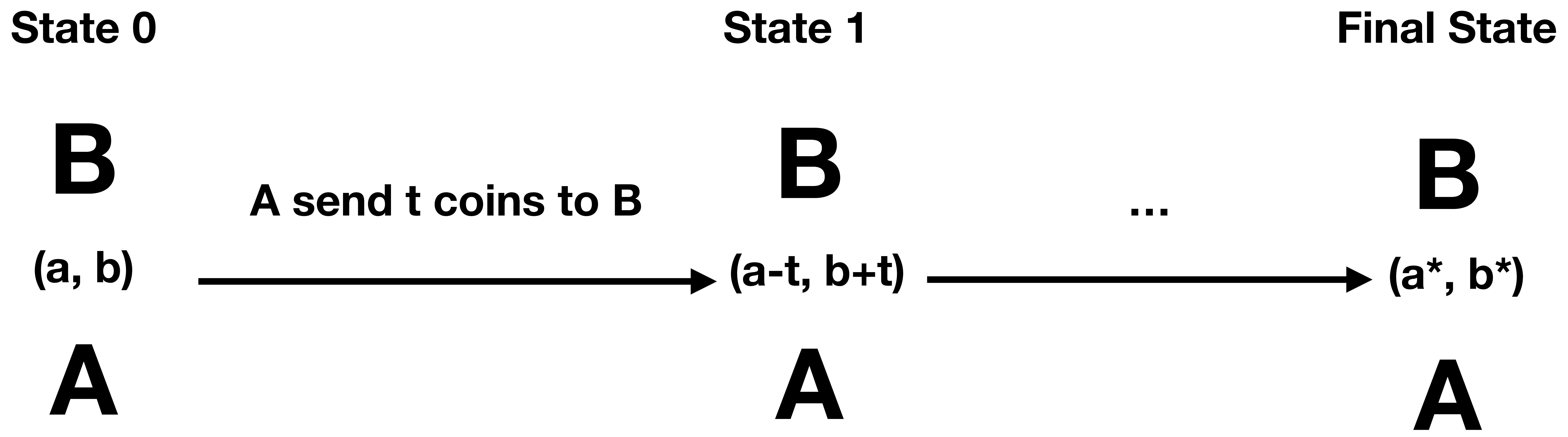
Payments Channel

TON vision

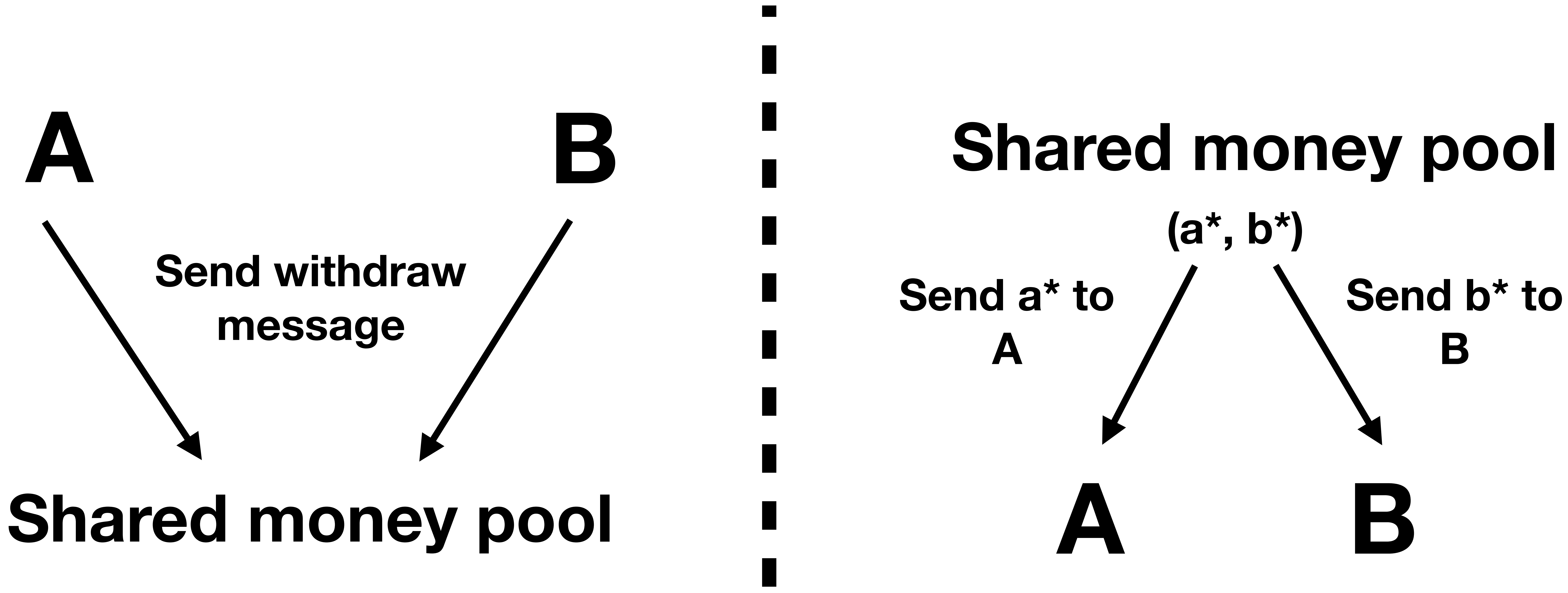
Made by Nick Kozlov
github.com/enorage ver 0.1


A
B
Send a coins
Send b coins
Shared money pool
Smart Contract
Blockchain

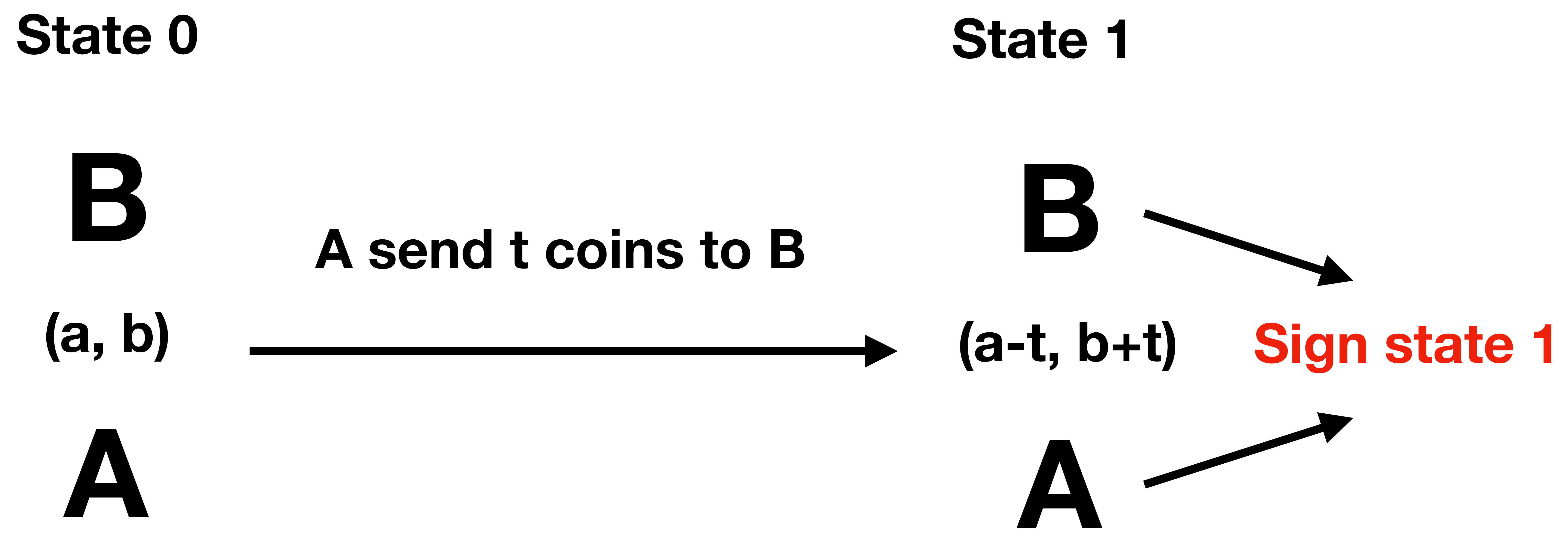
State of the Pool



Withdraw funds

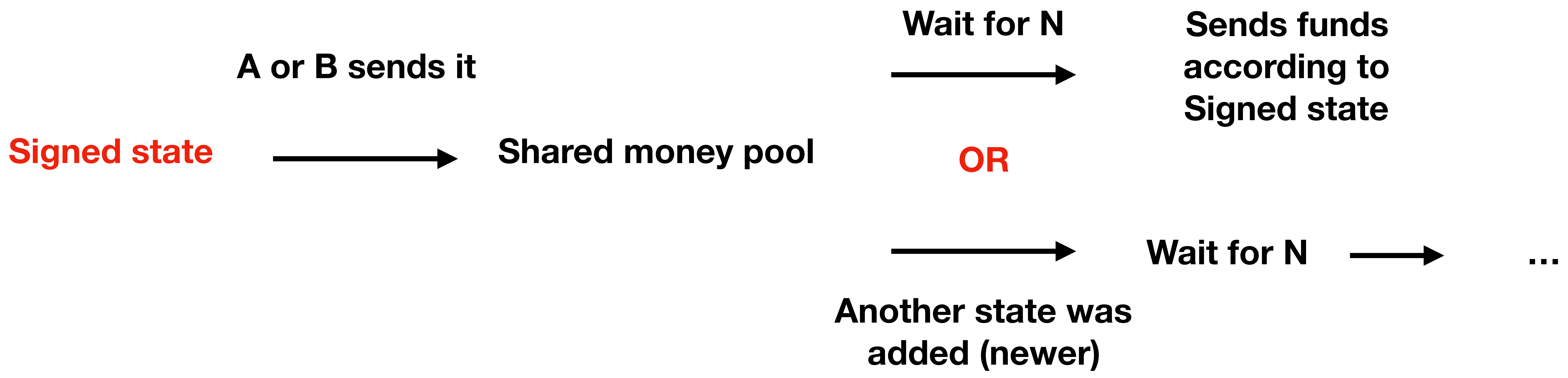


Trustless payment channels



Trustless payment channels

If channel was closed



Simple bidirectional synchronous trustless payment channel

Only A allowed to create State 0

State 0

$$\begin{tikzcd}
A \arrow[r, "\text{Create new state, sign it and send to B with signature}", "\text{Signature of A, new state data}"'] & B \arrow[r, "\text{Confirm it, sign and send copy of signature to A}", "\text{Signature of B}"'] & A
\end{tikzcd}$$

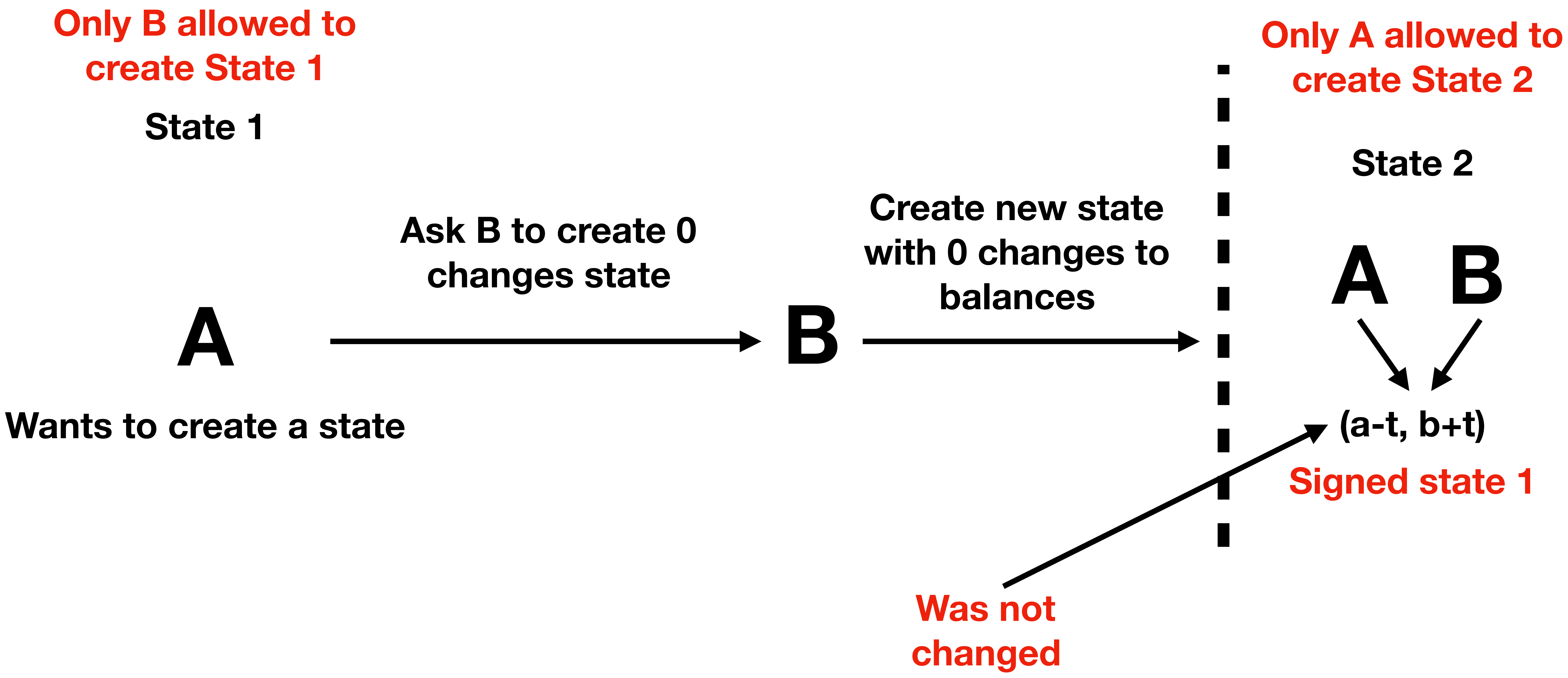
Only B allowed to create State 1

State 1

$$\begin{tikzcd}
A \arrow[dr] & & B \arrow[dl] \\
& (a-t, b+t) &
\end{tikzcd}$$

Signed state 0

Simple bidirectional synchronous trustless payment channel


Only B allowed to create State 1
State 1
A
Wants to create a state
Ask B to create 0 changes state
B
Create new state with 0 changes to balances
Only A allowed to create State 2
State 2
A
B
(a-t, b+t)
Signed state 1
Was not changed

Simple bidirectional synchronous trustless payment channel

Exit

Agreed

Invoke two-sided finalization method

$$\begin{tikzcd} A \arrow[dr] & & B \arrow[dl] \\ & \text{Shared money pool} & \end{tikzcd}$$

Final state

Signature A

Signature B

Disagreed

Invoke unilateral finalization

(its version of the final state, its final signature)

$$\begin{tikzcd} A \arrow[dr] & \text{or} & B \arrow[dl] \\ & \text{Shared money pool} & \end{tikzcd}$$

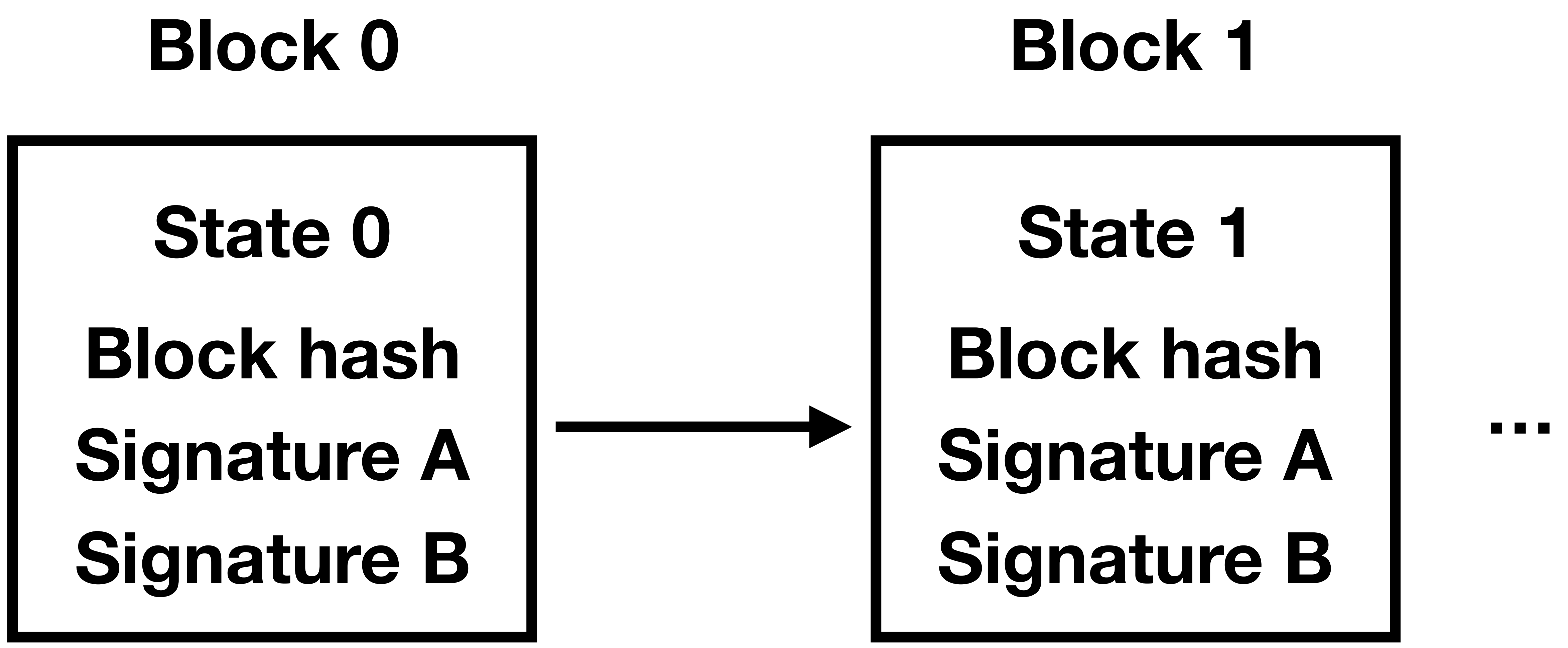
Simple bidirectional synchronous trustless payment channel

Disagreed exit

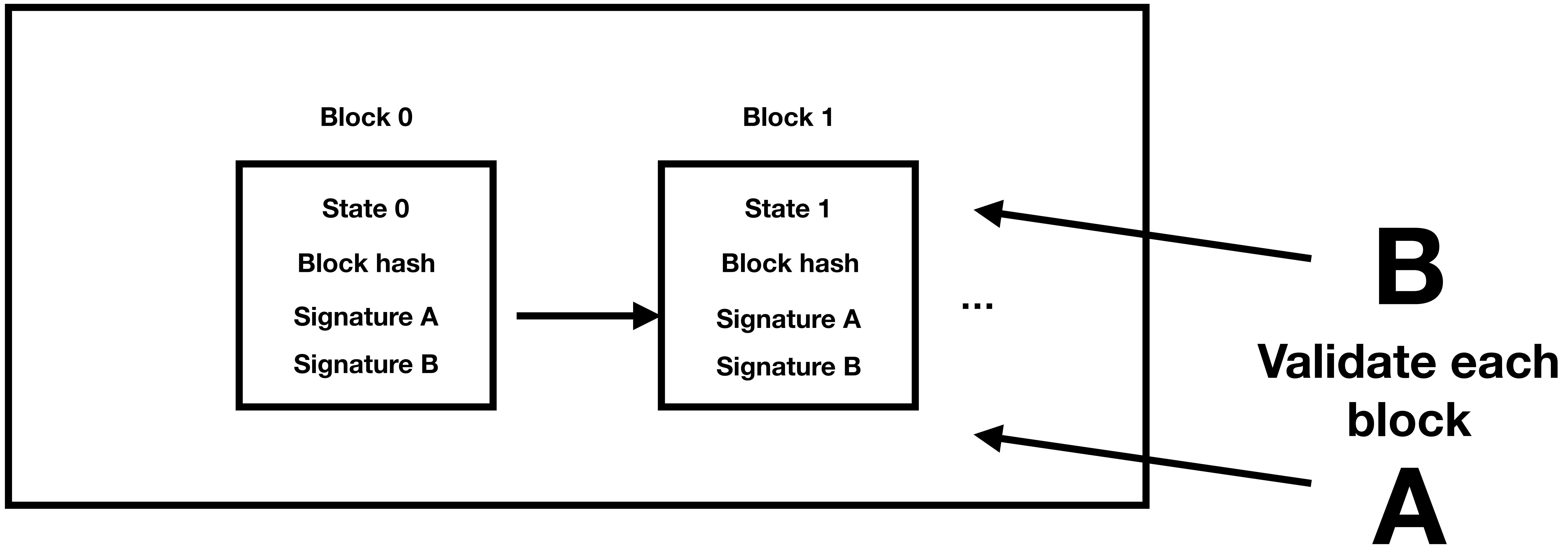
Shared money pool

When the other party submits its version and it turns out to be compatible with the already submitted version, the “true” final state is computed by the smart contract and used to distribute the money accordingly. If the other party fails to present its version of the final state to the smart contract, then the money is redistributed according to the only copy of the final state presented

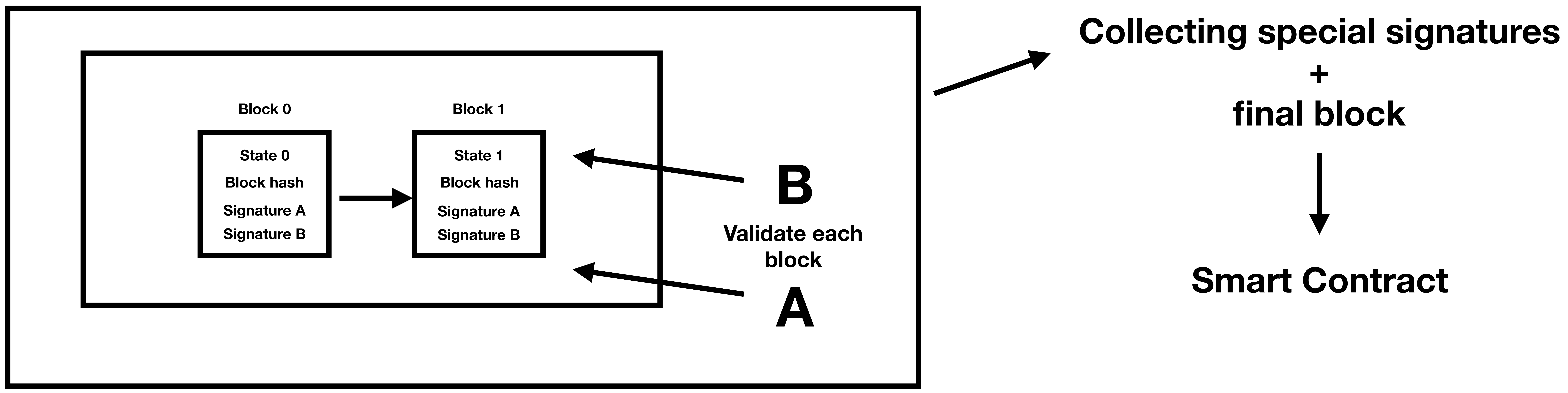
Synchronous payment channel as a simple virtual blockchain with two validators



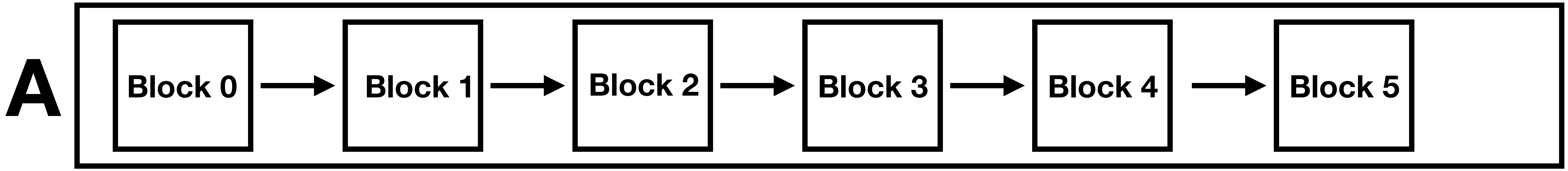
Synchronous payment channel as a simple virtual blockchain with two validators



Synchronous payment channel as a simple virtual blockchain with two validators

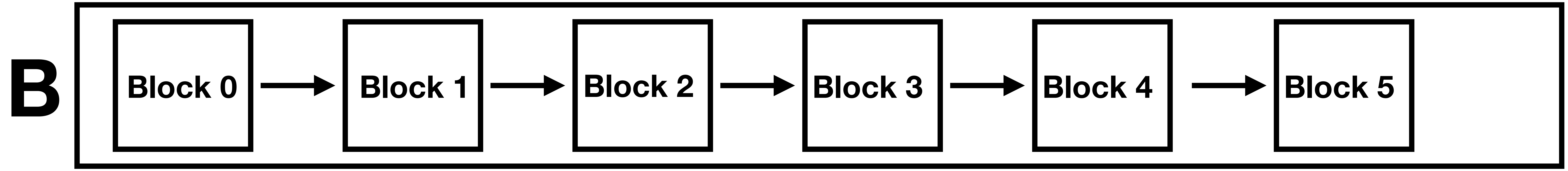


Asynchronous payment channel

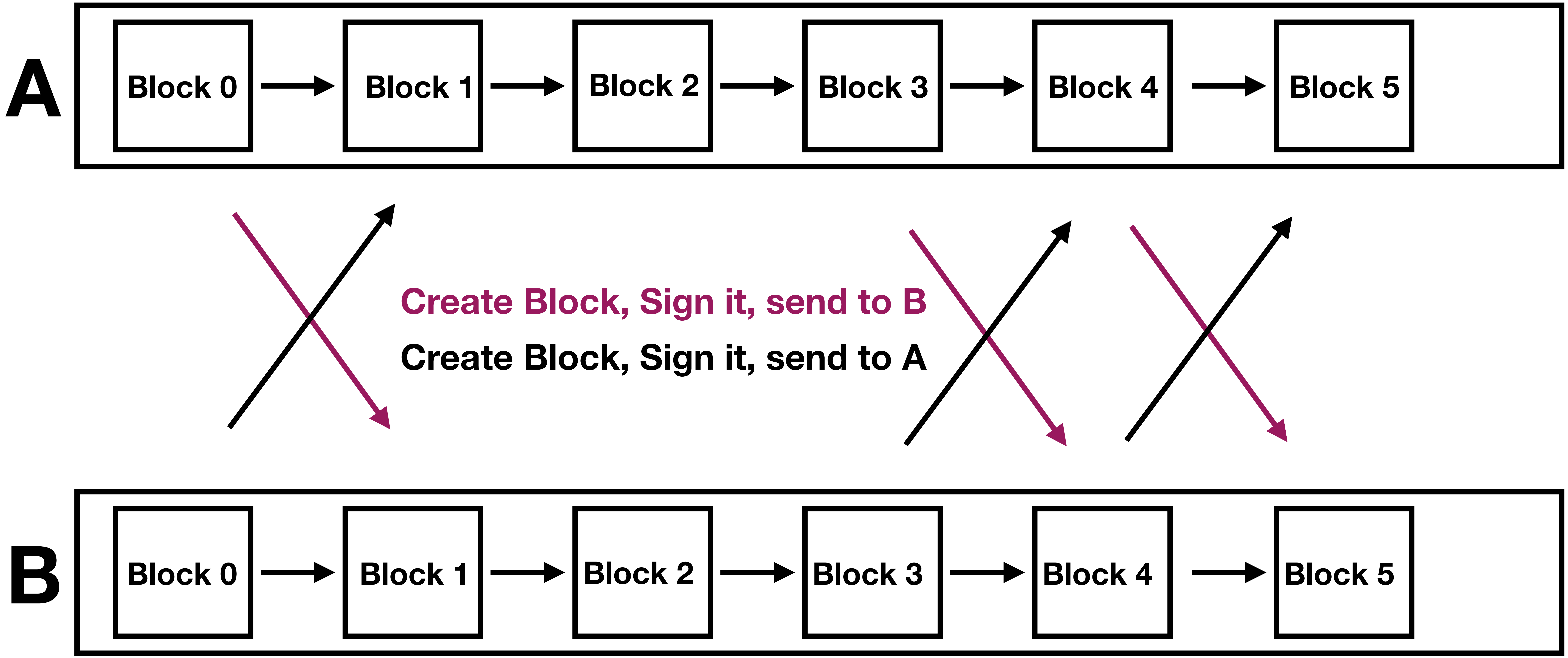


Each have it's own virtual blockchain
A create blocks at A chain

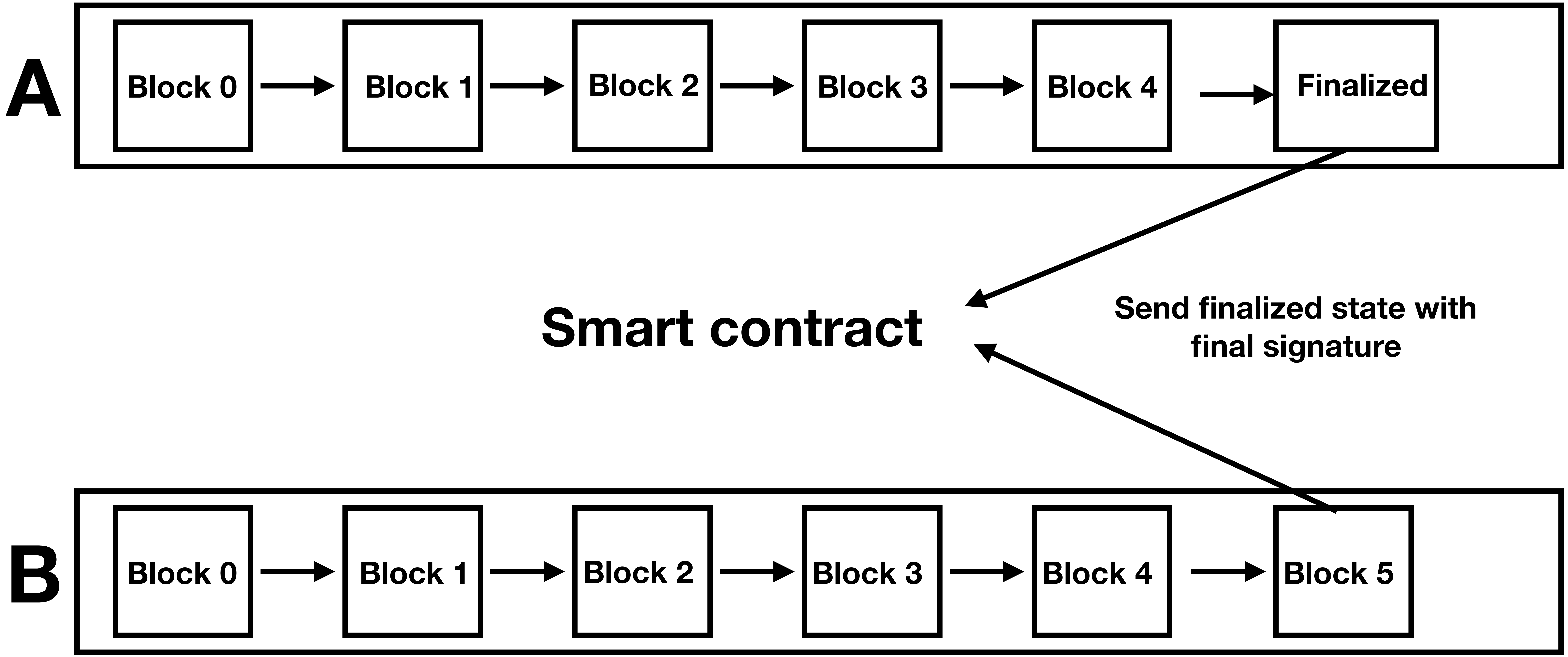
States
Block sequence number
Total amount from A to B
Block sequence number of B chain
Amount of money transferred from B to A



Asynchronous payment channel



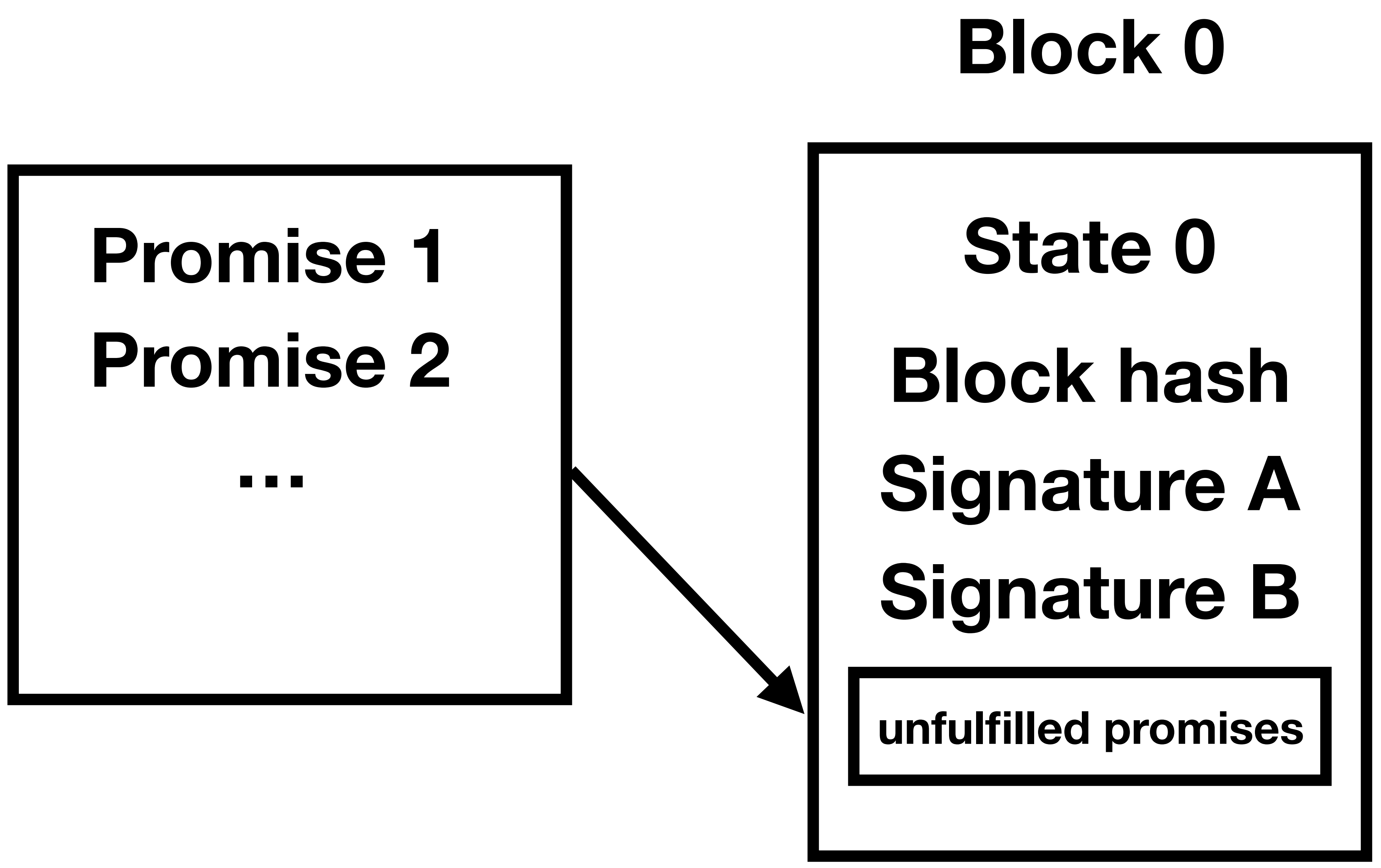
Asynchronous payment channel



Promises

$$A \longrightarrow B$$

A agrees to send c coins to B
only if
B can present some string u with Hash(u) = v for a known value of v



Lightning networks

$$\begin{tikzcd} A \arrow[r] & E \end{tikzcd}$$

A wants to send sum to E

A and E do not have open channel

$$\begin{tikzcd} A \arrow[r] & B & & C \arrow[r] & D \\ B \arrow[r] & C & & D \arrow[r] & E \end{tikzcd}$$

Chain money transfer

$$\begin{tikzcd}
 & A \arrow[rr] & & E & \\
A \arrow[r] & B & & C \arrow[r] & D \\
B \arrow[r] & C & & D \arrow[r] & E
\end{tikzcd}$$

A will send x coins to B and ask to send coins to C

A will create u and v = Hash (u)

Promise to pay x coins to B if a number u with hash v is presented

Promise contains v, but not u, which is still kept secret

Chain money transfer

A will need to present u to B

$$\begin{tikzcd} A \arrow[r] & B \end{tikzcd}$$

B creates a similar promise to C
B is not afraid of it, because A already have promise to B

$$\begin{tikzcd} A \arrow[r, "\text{promises}"] & B \\ B \arrow[r, "\text{promises}"] & C \end{tikzcd}$$

$$\begin{tikzcd} C \arrow[r, "\text{promises}"] & D \\ D \arrow[r, "\text{promises}"] & E \end{tikzcd}$$