Model Checking Blockchain-based System: A Case Study

Lung-Chen Huang, Naipeng Dong,

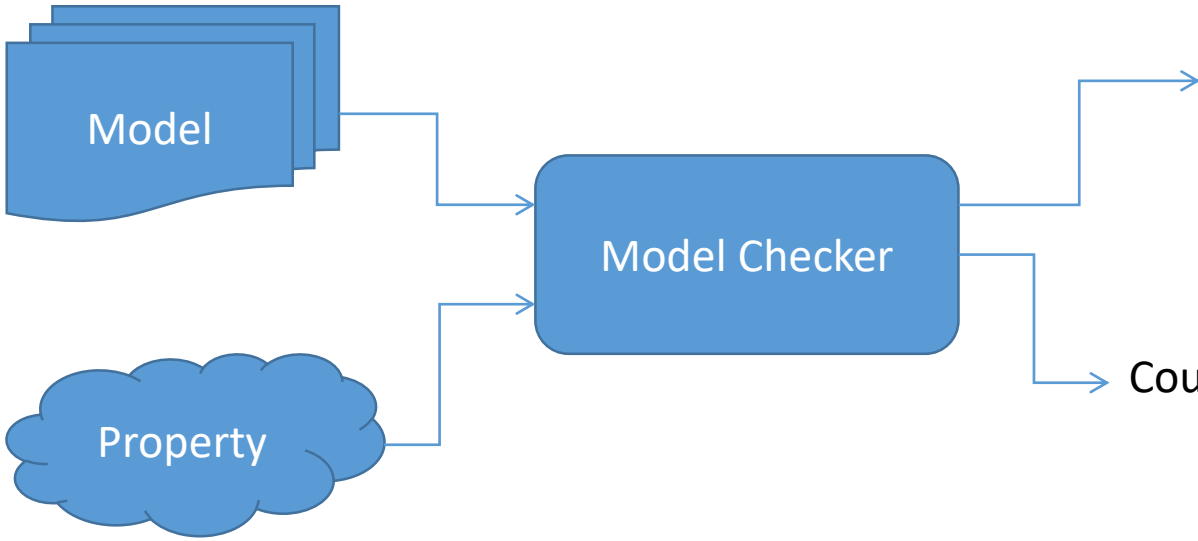
Guangdong Bai, Siau Cheng Khoo, Jin Song Dong

Outline

- Model Checking and PAT Introduction
- Blockchain-based Booking System and its Modeling
- Demo

Model checking

- Model checking: check whether a model satisfies a property by exhaustive searching.



Tool and modeling language used in this project

- Process Analysis Toolkit (PAT)
  - http://pat.comp.nus.edu.sg/
- Modeling language: CSP#, which is an extension of classic process algebra Communicating Sequential Processes (CSP)
  - Constants
    - *#define N 5*;
  - Variables of Type Bool, Integer, Arrays of integers
    - *var x: {0..10} = 5;*
    - *var x[N];*
  - Channels
    - *channel m 0;*
    - *channel m[M] channelBufferSize;*

Tool and modeling language used in this project (cont.)

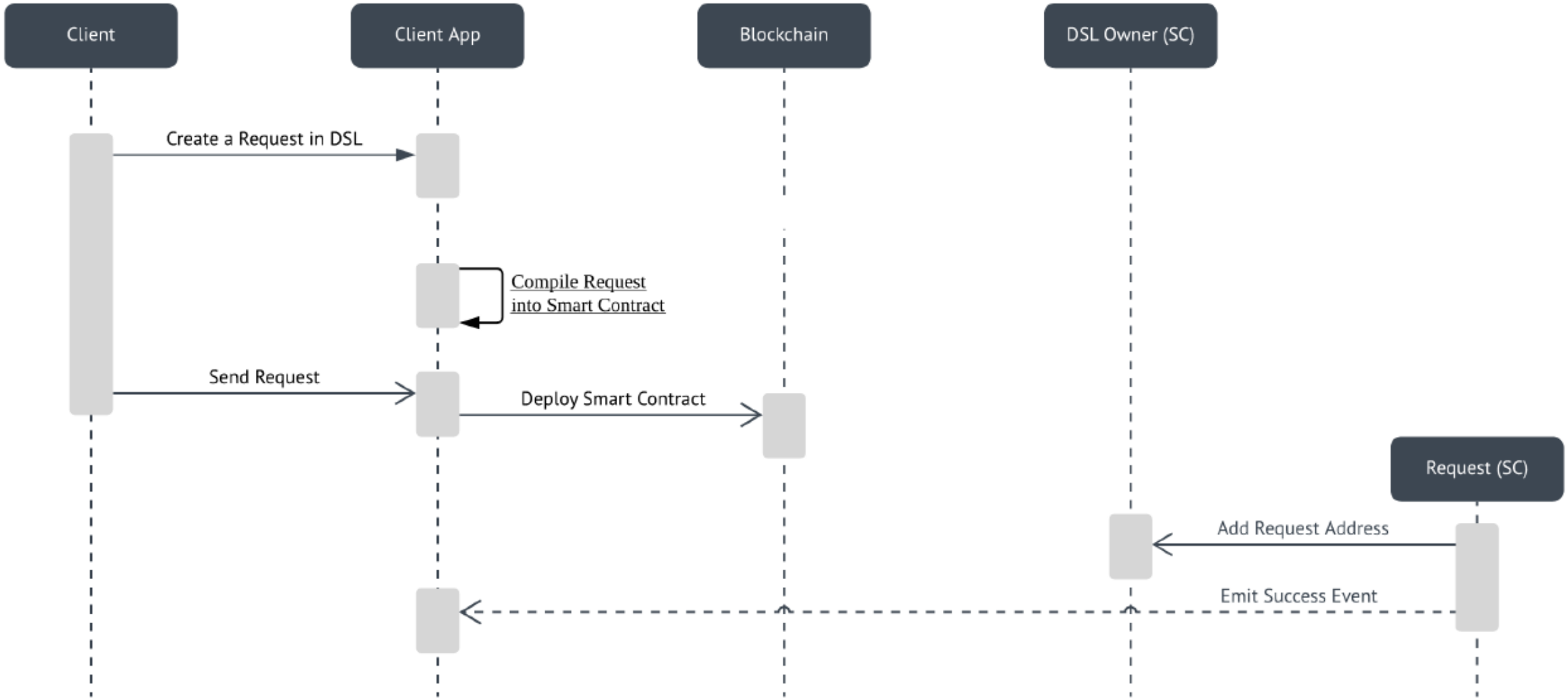
- Modeling language: CSP# (cont.)

Process Expression	Remarks
Stop	Do nothing
Skip	Termination, like Return
e{x:=1} -> P	Event prefixing
if(b) {P} else {Q}	Choice
P; Q	Sequential Composition
P || Q	Parallel Composition

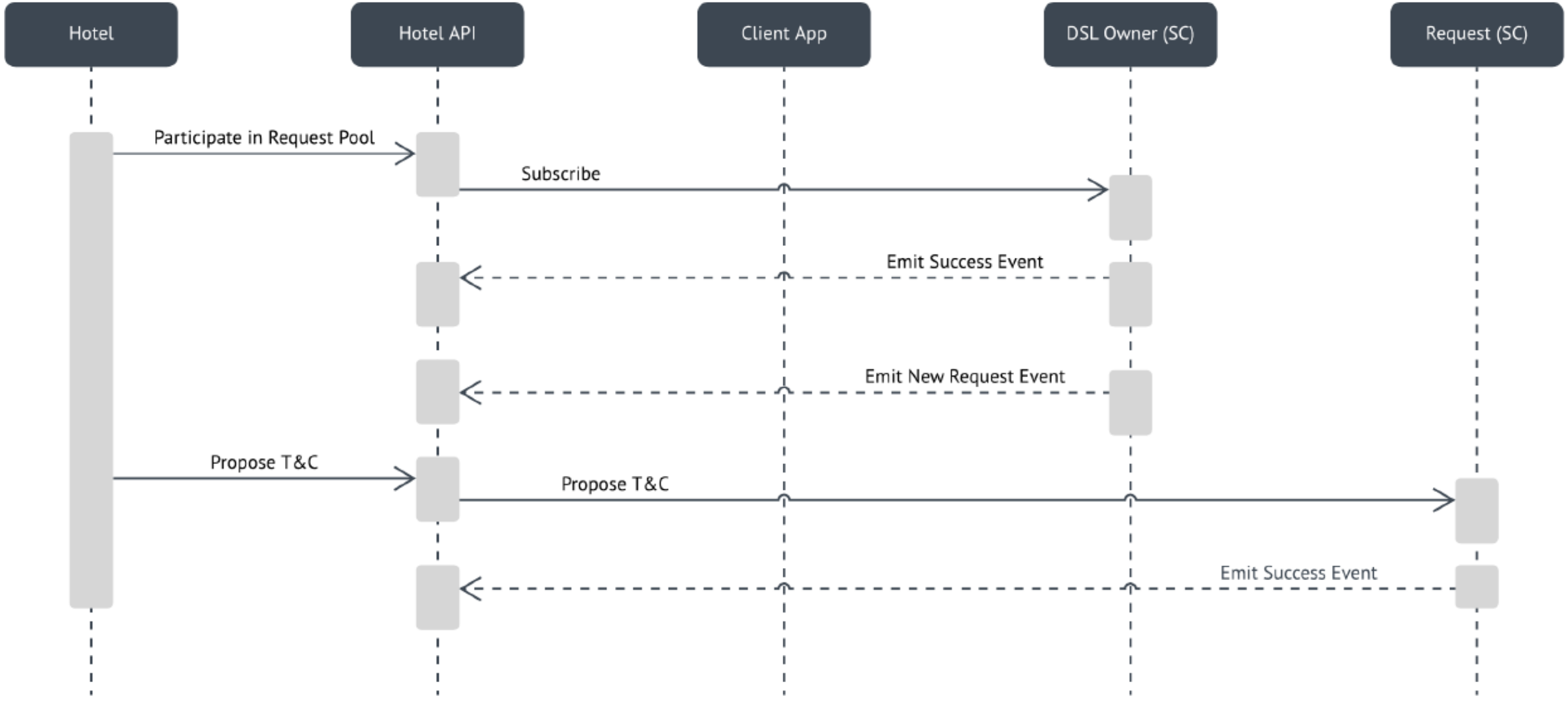
Blockchain-based hotel booking system

- Time and again, travelers experience difficulty in selecting hotels
  - Each time when a search result appears on screen, they still have hard time browsing through the choices and finding the best suited hotel room
- Hoteliers have extra cost to get known by travelers
- How blockchain can be beneficial and applicable?
  - A transparent ecosystem
  - The booking requests are similar (easy to contract)

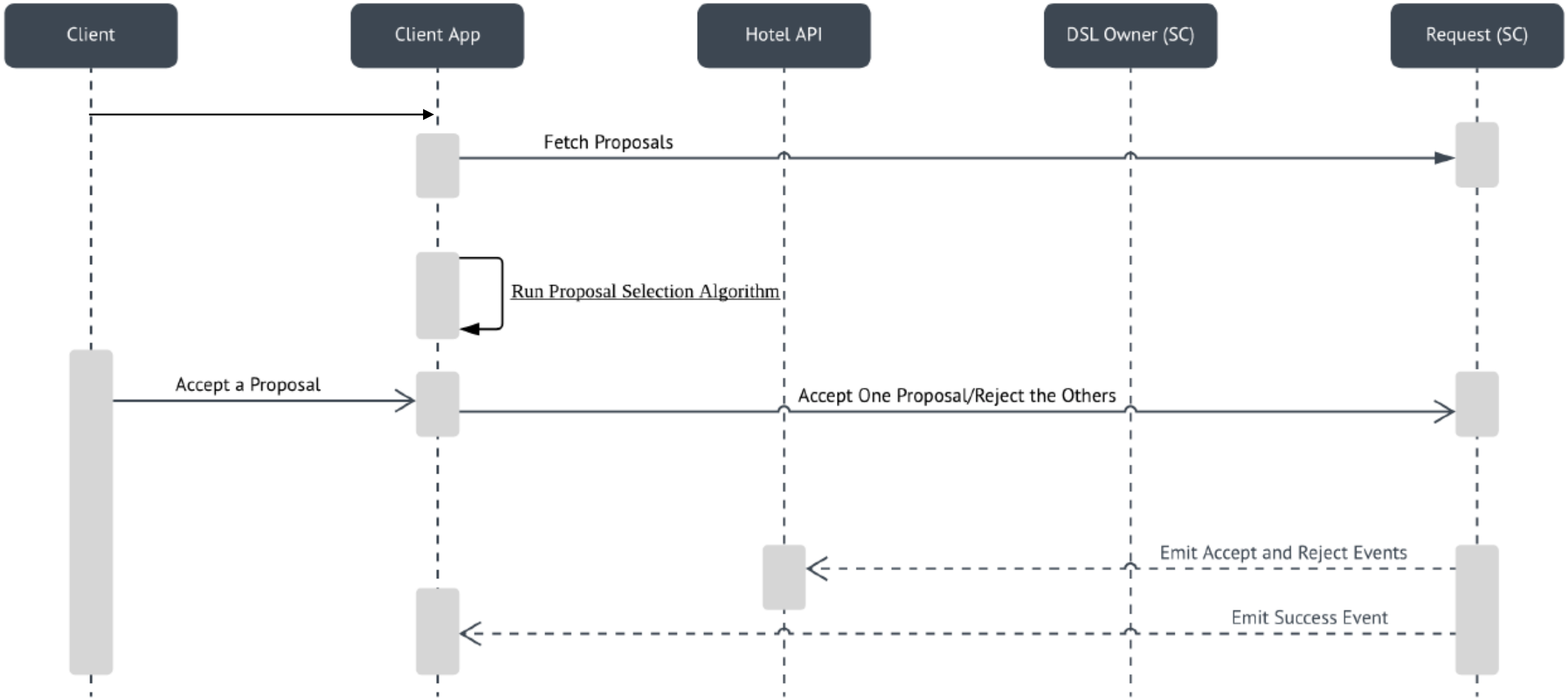
Make a request



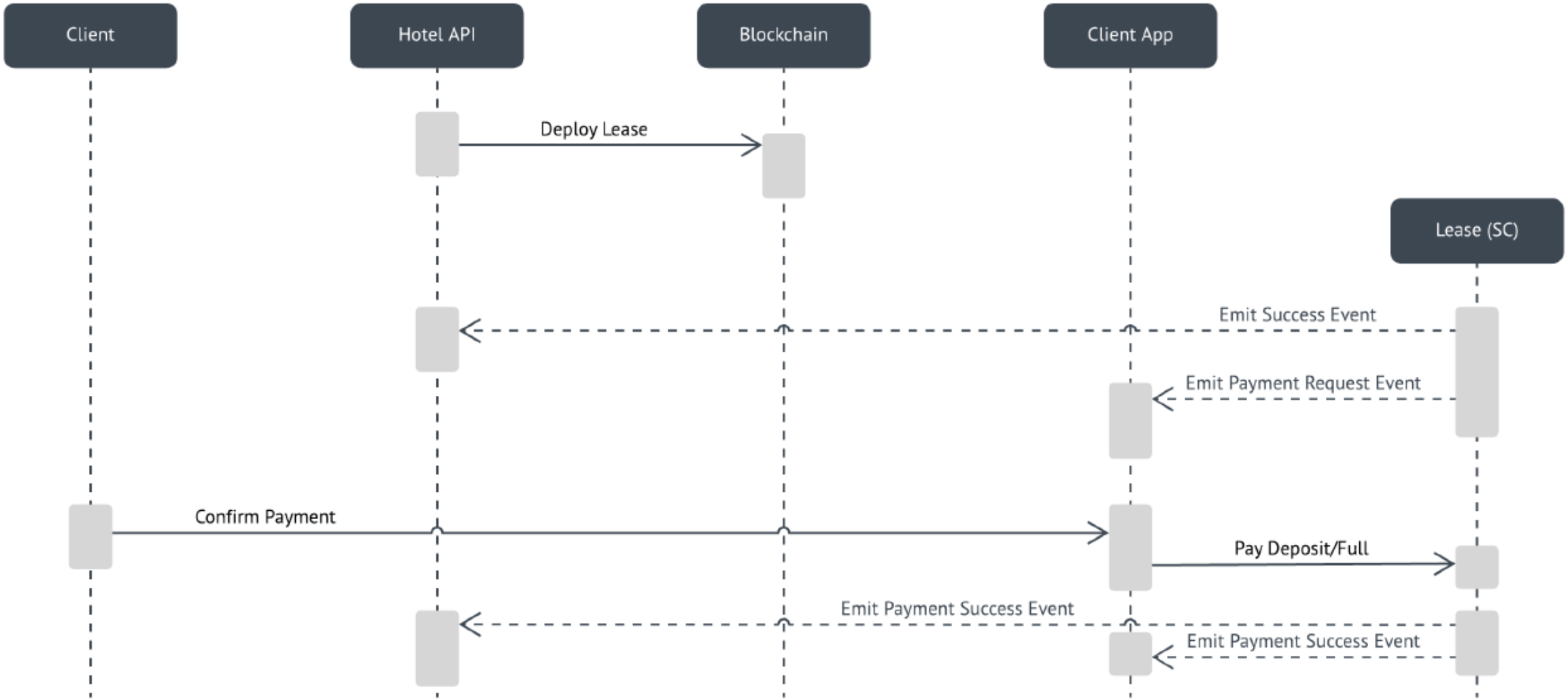
Offer a proposal



Accept/Reject proposals



Lease the room



Modeling – user issues a request

Traveler: any address that is willing to book a hotel room with the deployed request

```
User(addr, val, gas, gasPrice, function)
```

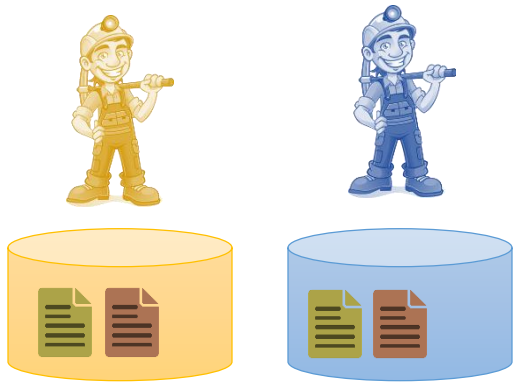
- addr → ID
- val, gas, gasPrice → Gas required for executing this contract
- function → ProposeFunction, SwitchFunction, FetchFunction, SettleFunction, …

```
= TxBrdcst(0, addr, val, gas, gasPrice, function);
```

Modeling – transactions are broadcast to miners

```
TxBrdcst(iter, addr, val, gas, gasPrice, function) = // broadcasting a transaction
        atomic{
                if (iter < M) {
                        mnet[iter]!addr.val.gas.gasPrice.function ->
                        TxBrdcst(iter+1, addr, val, gas, gasPrice, function)
                } else {
                        Skip
                }
        };
```

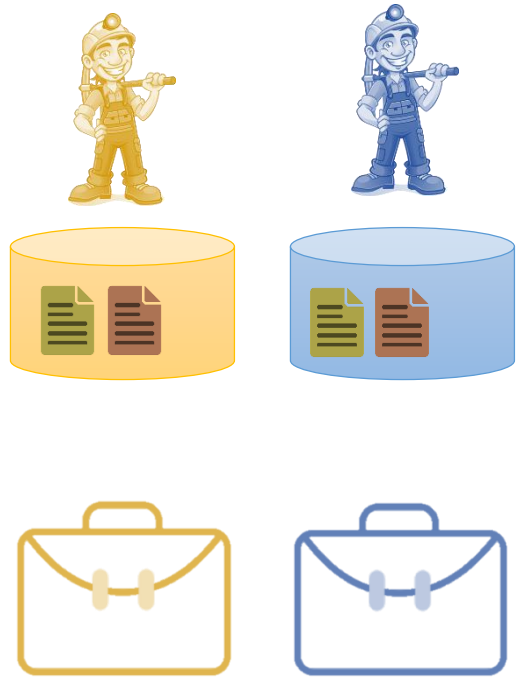
Number of miners



```
TxPool(i) = // inserting new transactions into the pool
        mnet[i]?addr.val.gas.gasPrice.function ->
        if (userWallet[i][addr] >= (val + (gas * gasPrice)) && gas > 0) { // a checkpoint on
whether or not the sender has enough funds and gas before a miner accepts the transaction
                TxInsert(i, addr, val, gas, gasPrice, function)
        } else {
                TxPool(i)
        };
```

Omitted, which insert the transaction into the miner-owned pools

Modeling – block construction



```
TxBlock(i) = // including available transactions from the pool
    if (inPoolPtr[i] < poolSize) {
        if (!call(cempty, datach[i])) {
            datach[i]?newcomer ->
            if (poolTxGas[i][inPoolPtr[i]] == 0) {
                if (blkSize[i] >= 1) {
                    Miner(i, 0)
                } else {
                    TxBlock(i) }
            } else {
                txInclude{
                    blkTxAddr[i][bIter] = poolTxAddr[i][pIter];
                    blkTxVal[i][bIter] = poolTxVal[i][pIter];
                    blkTxGas[i][bIter] = poolTxGas[i][pIter];
                    blkTxGasPrice[i][bIter] = poolTxGasPrice[i][pIter];
                    … …
                    inPoolPtr[i]++;
                    blkSize[i]++;
            }
            … …
    };
```

Modeling – mining

```
Miner(i, iter) = // start mining the block where
transactions were collected
        if (iter < blkSize[i]) {
                LockUp(i, iter)
        } else {
                BlkUpdate(i, 0)
        };

LockUp(i, iter) = // to lock up the total value including the gas to execute the transaction from a user
        weiLockUp{
                var addr = blkTxAddr[i][iter];
                var val = blkTxVal[i][iter];
                var gas = blkTxGas[i][iter];
                var gasPrice = blkTxGasPrice[i][iter];
                var lockedTotal = val + (gas * gasPrice);
                userWallet[i][addr] = userWallet[i][addr] - lockedTotal;
                lockedWallet[i] = lockedTotal;
        } ->
        TxExec(i, iter);
```

①

Modeling – contract execution

```
TxExec(i, iter) = // acting like invoking predefined functions in the
smart contract; here just dispatch the execution for each of the functions
        case {
                blkTxFunction[i][iter] == SwitchFunction:
                        … …
                blkTxFunction[i][iter] == ProposeFunction:
                    if (contractSwitch[i] == on) {
                        GasConsume(i, iter, UPDATE) || Execution(i, iter)
                    } else { LockedReturn(i, iter, false)   }
                blkTxFunction[i][iter] == SettleFunction:
                        … …
                blkTxFunction[i][iter] == FetchFunction:
                        … …
        };

BlkDetect(i, iter, success) = // before executing the next transaction, check whether there is any
block from another miner. Under assumption that the miner chooses the first arrival block
        if (call(cempty, bnet[i])) { Miner(i, iter+1)
        } else { bnet[i]?j.newBlockNum.blockid -> BlkAppend(i, j, newBlockNum, blockid) };
```

Modeling – commit a mined block

```
Miner(i, iter) = // start mining the block where
transactions were collected
        if (iter < blkSize[i]) {
                LockUp(i, iter)
        } else {
                BlkUpdate(i, 0)
        };

BlkUpdate(i, iter) = // update the globally shared state
        if (iter < blkSize[i]) {
                update{
                        block[blockUid][iter] = pendUid[i][iter];
                        txFromAddr[pendUid[i][iter]] = pendFromAddr[i][iter];
                        txToAddr[pendUid[i][iter]] = pendToAddr[i][iter];
                        txOp[pendUid[i][iter]] = pendOp[i][iter];
                        … …      } ->
                BlkUpdate(i, iter+1)
        } else {
                reward{minerCoinbase[i] = minerCoinbase[i] + succAppendPrice; rewardCount++;} ->
                BlkBrdcst(i, 0, blockNum[i], blockUid)
        };
```

②

Modeling – append to blockchain

```
BlkAppend(i, j, newBlockNum, blockid) = // append the latest block and update the state in the blockchain
        append{
                chain[i][newBlockNum] = blockid;
                blockNum[i]++;
                blkSize[i] = blkSize[j];
        } ->
        ChainUpdate(i, 0, blockid);
```

```
ChainUpdate(i, iter, blockid) = // update their own chain
        case {
                txOp[block[blockid][iter]] == FetchFunction:
                … …
                txOp[block[blockid][iter]] == SwitchFunction:
                … …
                txOp[block[blockid][iter]] == ProposeFunction:
                    propose{ … …  } -> ChainUpdate(i, iter+1, blockid)
                txOp[block[blockid][iter]] == SettleFunction:
                    settle{ … …  } ->  ChainUpdate(i, iter+1, blockid)

                }
```


DEMO

Experiment results

- C = 5, M = 2 (5 users, 2 miners)

Assertion	States	Transitions	Time	Result
ProposerExecution deadlockfree	8328	18823	1.76	**Valid**
ProposerExecution |= []! GasRunOut	13413	30563	2.68	**Valid**
ProposerExecution reaches sameBlockNumEventually	285	336	0.04	**Valid**
ListenerExecution |= proposalReceived → listenerReceiving	13413	30563	2.76	**Valid**
UnavailableExecution |= []! proposalReceived	13413	30563	2.73	**Valid**
NotOwnerExecution |= []! fetch	4537	11940	0.78	**Valid**
MultipleUsersExecution reaches sameBlockNumEventually	504	521	0.05	**Valid**
SettlementExecution reaches receiveSettlement	362	605	0.06	**Valid**

Experiment results

- C = 5, M = 10 (5 users, 10 miners)

Assertion	States	Transitions	Time	Result
ProposerExecution deadlockfree	6774622	68280939	3647.43	**Incomplete**
ProposerExecution |= []! GasRunOut	169626	2165266	112.52	**Incomplete**
ProposerExecution reaches sameBlockNumEventually	297717	415836	21.45	**Valid**
ListenerExecution |= proposalReceived → listenerReceiving	230685	2897974	415.428	**Incomplete**
UnavailableExecution |= []! proposalReceived	169929	2168902	111.65	**Incomplete**
NotOwnerExecution |= []! fetch	317081	1511802	189.67	**Incomplete**
MultipleUsersExecution reaches sameBlockNumEventually	10312	10425	0.98	**Valid**
SettlementExecution reaches receiveSettlement	3313604	32332261	1607.98	**Incomplete**

Thank You!