# Security Case Study with the Downsizing Game

Théo Dubourg

Real Life Security Seminar - 2014

# Goal of the Work

1. We want to study a security protocol
2. **Implementation-oriented work / software engineering exercise**
3. Case Study: The Downsizing Game
4. Study how secure we can implement the game
5. Use the case study to learn and point to security issues that arise throughout implementation
6. Comparison of the implementation against coding security guidelines

# The Downsizing Game

## *Simplified*

1. A group of players (for us: 3)
2. 1M money given at the beginning
3. Maximize their profit
4. Vote for *other* players
5. Winner if the highest score
6. At the end, give back the starting money
7. A *judging party* or *game master* controls the game

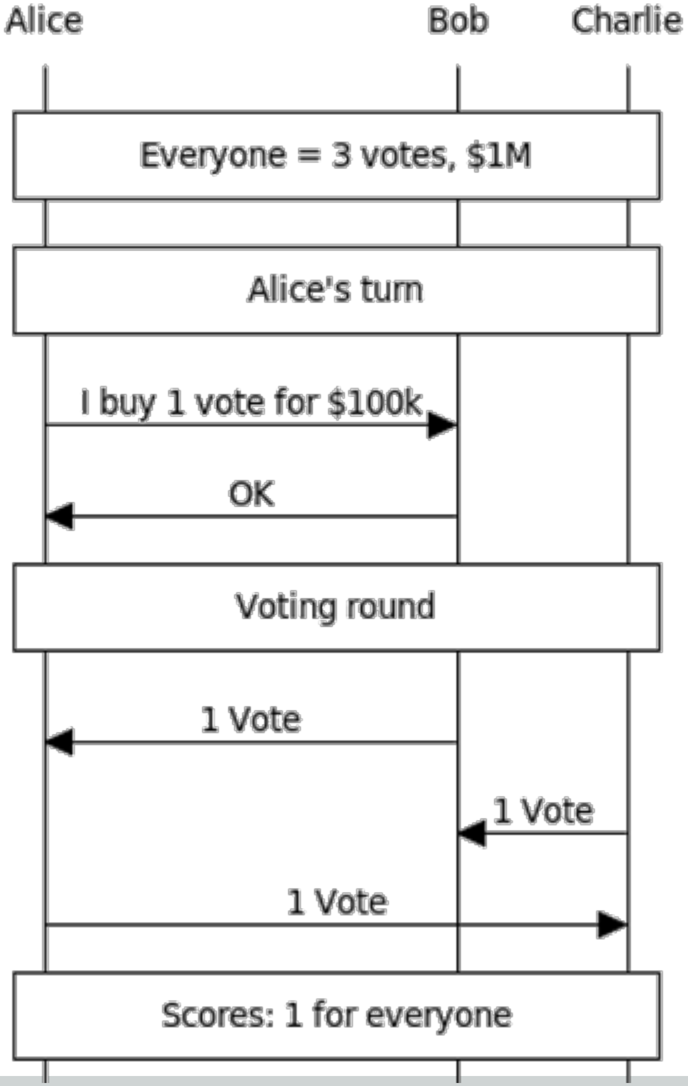
# The Downsizing Game

## *Game Example 1/4*

- 3 players

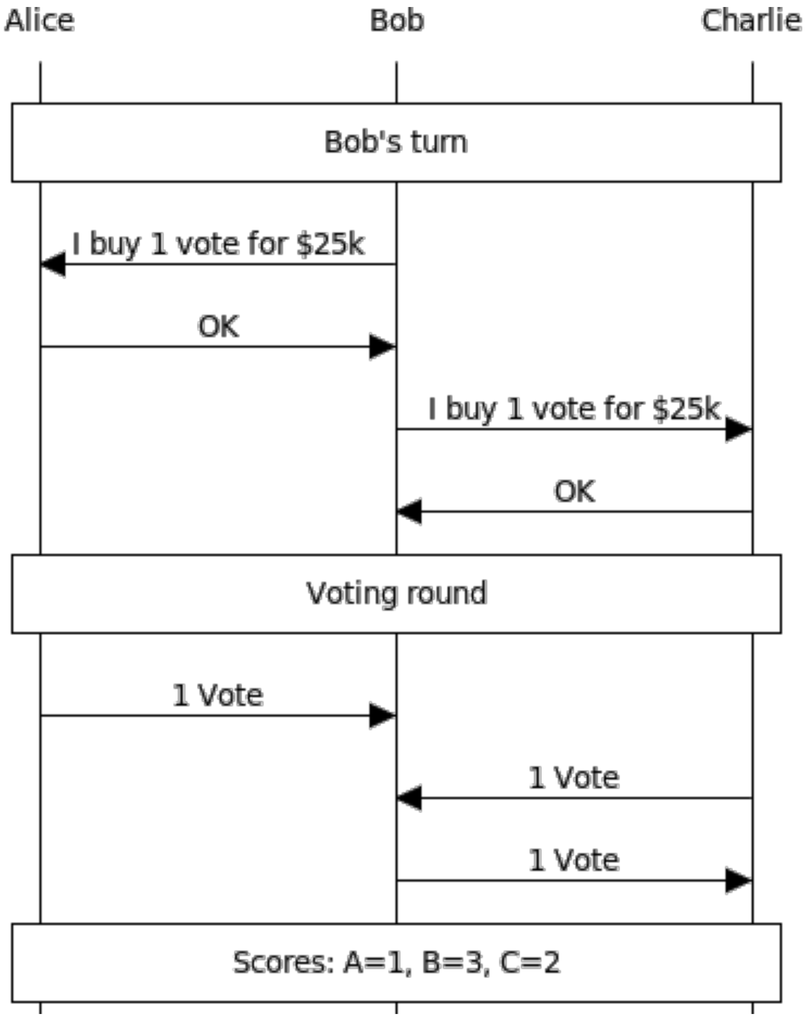
- 6 rounds

- 3 voting rounds



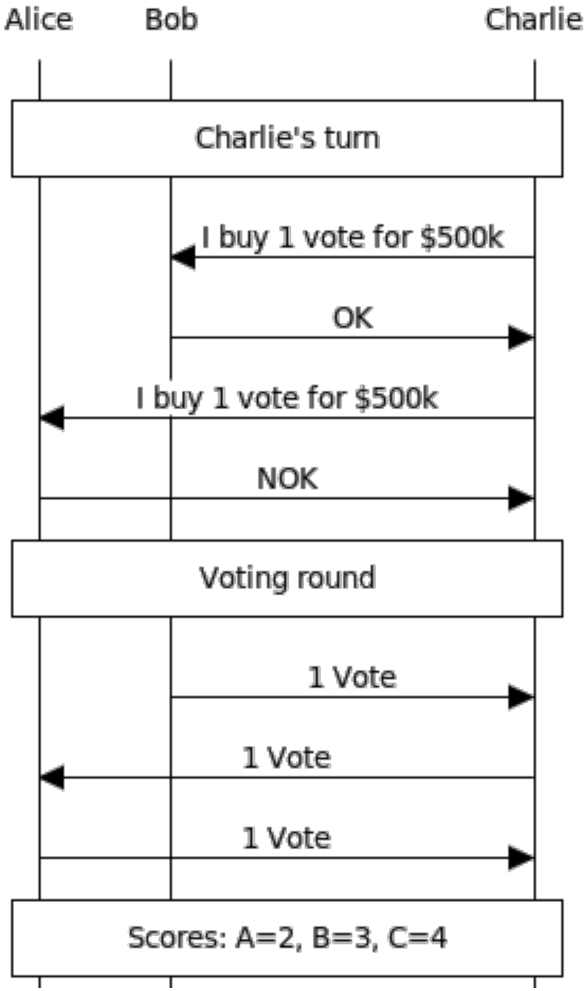
# The Downsizing Game

## *Game Example 2/4*



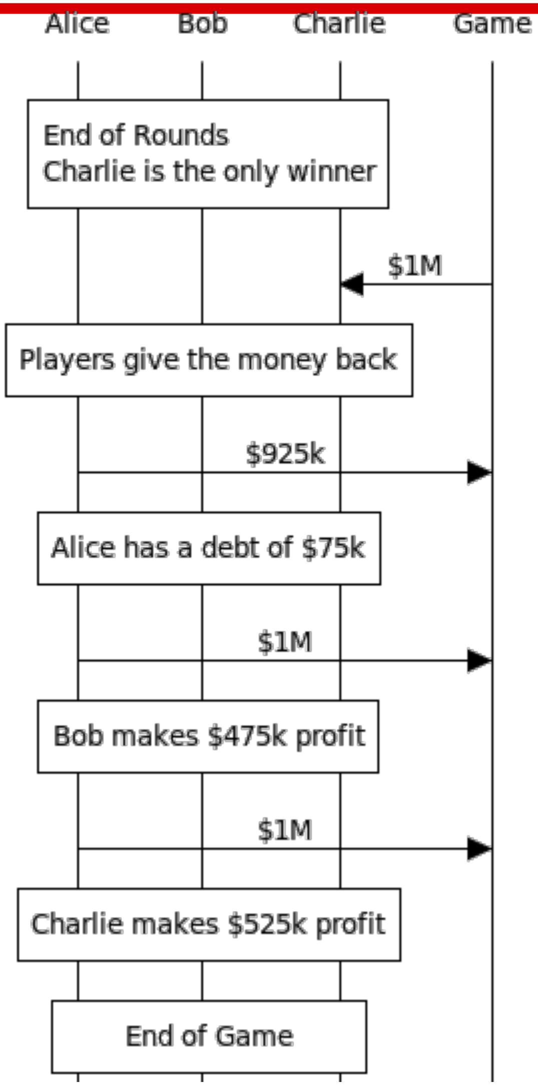
# The Downsizing Game

## *Game Example 3/4*



# The Downsizing Game

## *Game Example 4/4*



Alice

Bob

Charlie

Game

End of Rounds
Charlie is the only winner

$1M

Players give the money back

$925k

Alice has a debt of $75k

$1M

Bob makes $475k profit

$1M

Charlie makes $525k profit

End of Game

# The Downsizing Game

## *Our Instance's Rules*

- Fixed length of 1,000 rounds
- 3 players
- Participating **only once** (cannot purge debt using the game again)
- Tradable *resources*: votes, score and dollars
- At start: $1M (to be returned at the end), 10 votes
- Starting score = 0
- Remainder of the money can be kept
- Cannot give back the money = debt contracted
- Two players can trade resources using *transactions*
- *A judging party*
    - enforces the game's rules (cheating prevention)
    - manages transactions (validity, cheating prevention)

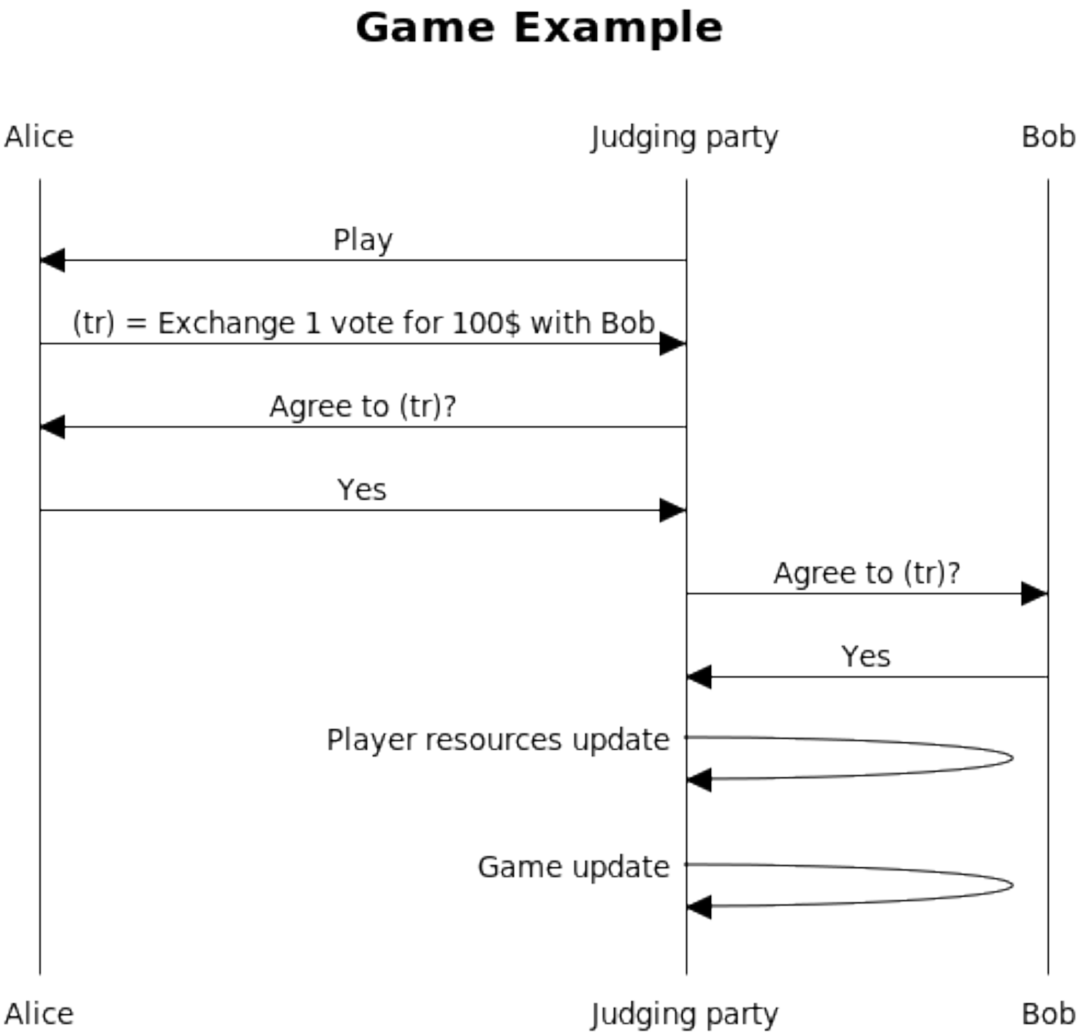
# The Downsizing Game

## *Our Instance’s Rules (continued)*

- 10 voting rounds (every 100 rounds)
- All players must vote on voting rounds
- Players must cast exactly 1 vote at every voting round
- Players can not vote for themselves
- 1 vote received = scored increased by 1
- Highest score at the end = winner = $1M prize
- If a player is banned (cheating), starting money must be returned

# The Downsizing Game

## *Our Instance's Security Protocol*



# Functional Requirements & Definitions: Rounds

1. Players play turn-by-turn
2. *Current player* allowed to contact judging party
3. Round passes when
   a. Transaction applied
   b. Changing current player
   c. Applying the result of a voting round

# Functional Requirements & Definitions: Transactions

1. Unidirectional vs. Bidirectional
2. Immediate vs. Scheduled (“delayed”)
3. Fixed amount
4. Transactions history log

# Functional Requirements & Definitions: Transactions Validation

1. Immediate transactions
   a. Immediate validation or refusal
2. Scheduled transaction
   a. Pre-validation / acceptance
   b. On deadline: transactions history log analysis to check for fulfilment of the agreed amount to have been transferred
   c. Subtransactions with “parent transaction” id
3. Not fulfilled: Cheating attempt (ban)

# Functional Requirements & Definitions: Voting rounds

1. Each player casts exactly 1 vote
2. Valid votes are registered as special transactions
    a. Those transactions can only be instantiated and applied by the judging party, not players

# Functional Requirements & Definitions: Voting Promises Transactions

1. Scheduled transaction
2. Fulfilment checked as normal for scheduled transactions, using “voting transactions”

# Security Requirements & Definitions: Game & Judging party protection

1. Judging party should only execute actions from the *current player*, no other players.
2. We need to *authenticate* players
3. Judging party *neutrality*
4. *Game state* alteration
5. Resources accounting protection

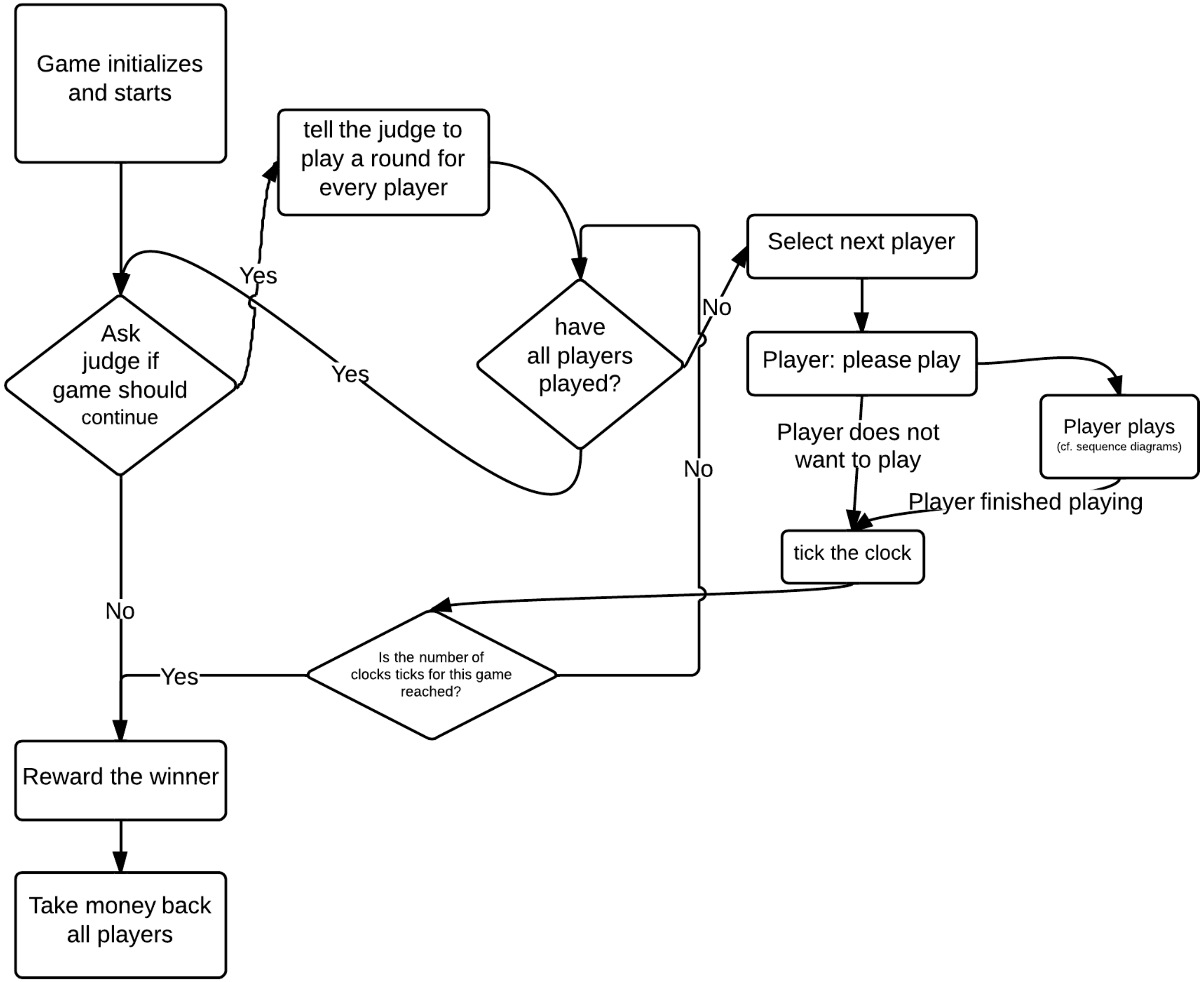
# Security Requirements & Definitions: Transactions

1. *Immediate* transactions should be *atomic*
2. Scheduled transactions should be *completed* before the *deadline*
3. Transactions should only be approved if both players, **authenticated**, agreed on it.

# Other related issues

1. Players that are banned might owe some resources (pending scheduled transactions)
   a. Divide the remaining resources of the banned player proportionally to what it owed to you
2. Balances are not accessible at all by players, they know their initial balance and then have to track it themselves → Avoid protection mechanism on the balances.
3. Some input validation is needed for transactions (negative amounts, banned players as the other trader, invalid deadlines…)

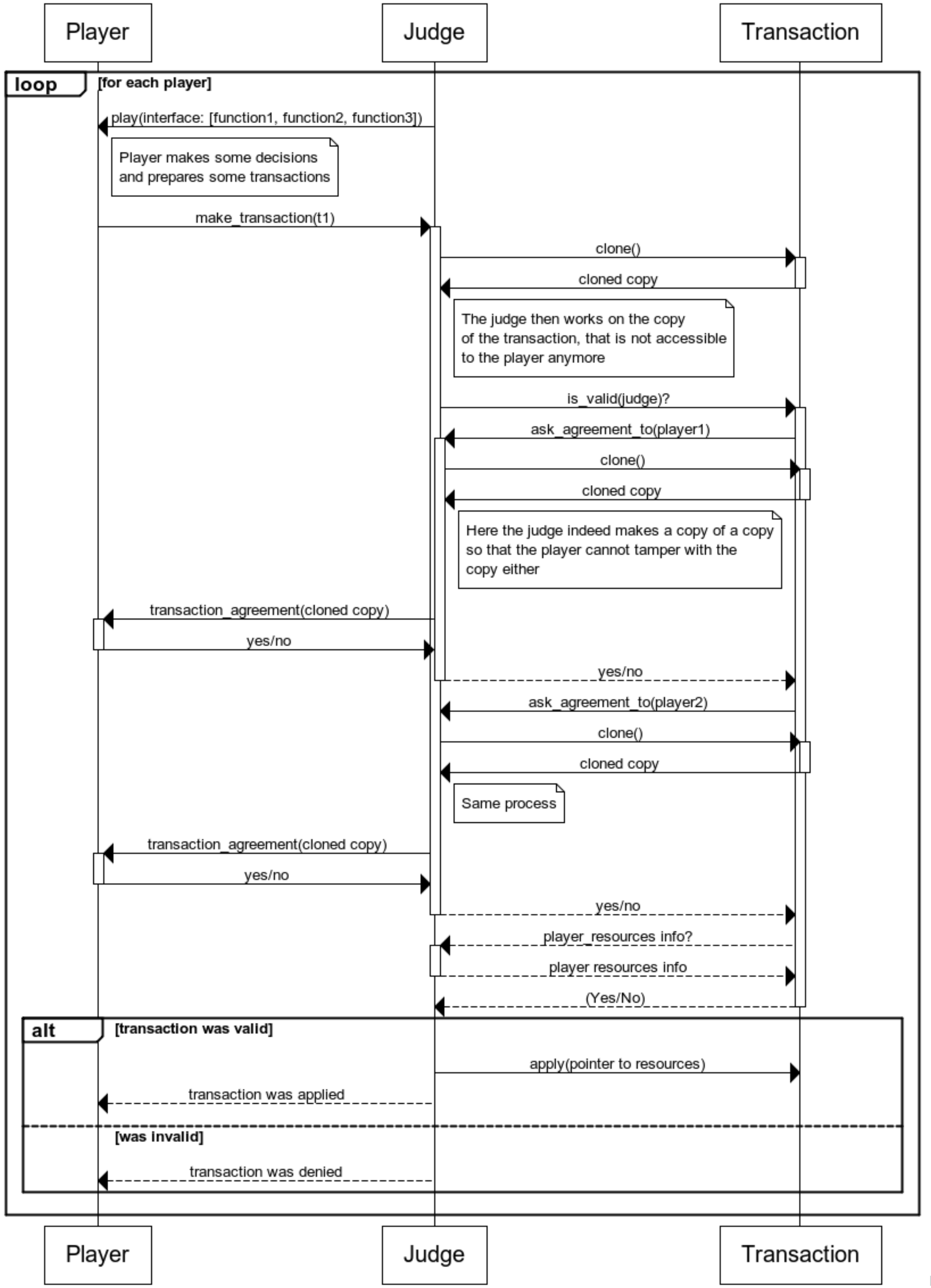
# Game System Overview


Game initializes and starts
tell the judge to play a round for every player
Select next player
Yes
Ask judge if game should continue
Yes
have all players played?
No
Player: please play
Player plays
(cf. sequence diagrams)
Player does not want to play
No
Player finished playing
tick the clock
No
Yes
Is the number of clocks ticks for this game reached?
Reward the winner
Take money back all players

THE

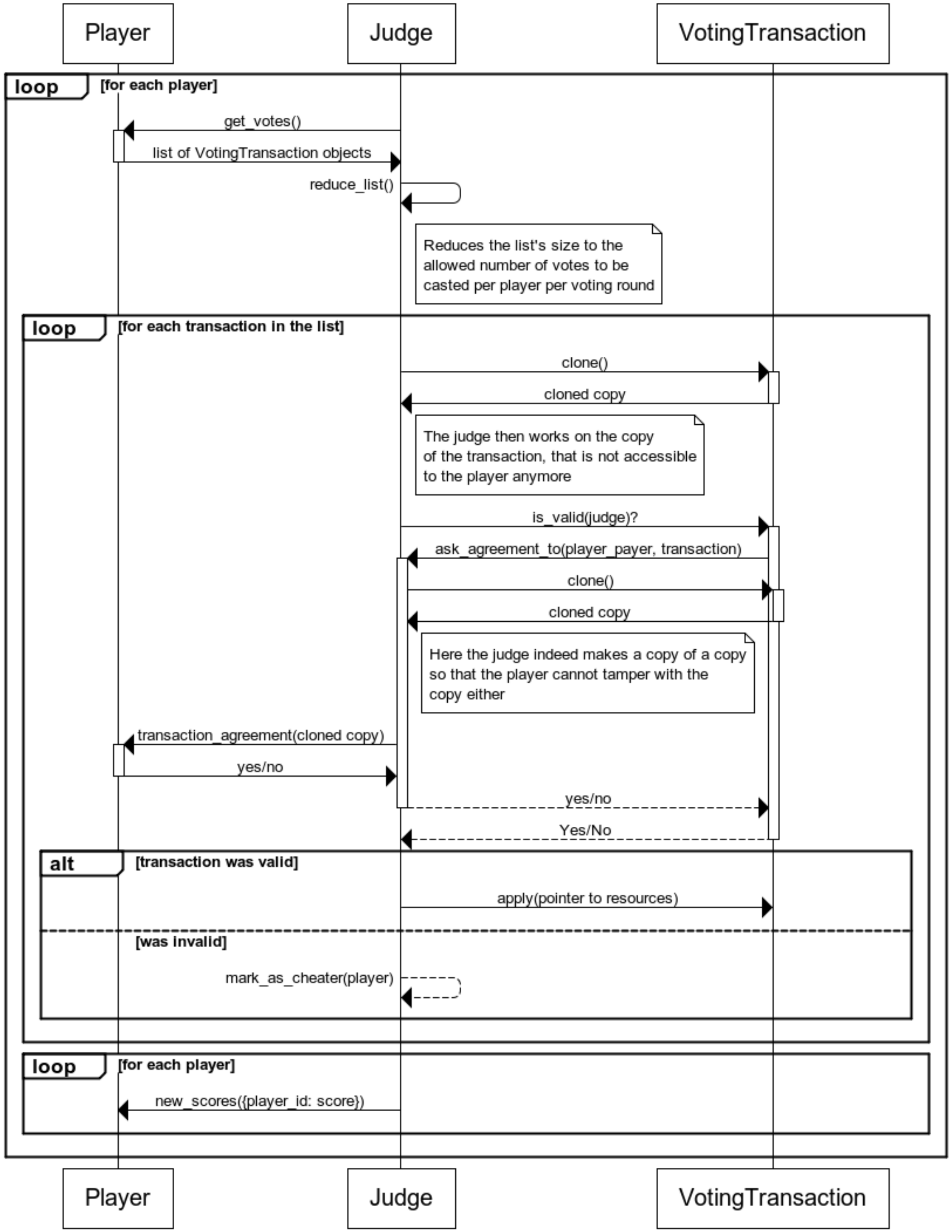
END

## Non-voting round: Transaction validation and application process


Player
Judge
Transaction
loop
[for each player]
play(interface: [function1, function2, function3])
Player makes some decisions
and prepares some transactions
make_transaction(t1)
clone()
cloned copy
The judge then works on the copy
of the transaction, that is not accessible
to the player anymore
is_valid(judge)?
ask_agreement_to(player1)
clone()
cloned copy
Here the judge indeed makes a copy of a copy
so that the player cannot tamper with the
copy either
transaction_agreement(cloned copy)
yes/no
yes/no
ask_agreement_to(player2)
clone()
cloned copy
Same process
transaction_agreement(cloned copy)
yes/no
yes/no
player_resources info?
player resources info
(Yes/No)
alt
[transaction was valid]
apply(pointer to resources)
transaction was applied
[was invalid]
transaction was denied
Player
Judge
Transaction

www.websequencediagrams.com

# Voting round: Votes casting and voting transaction validation.



Reduces the list's size to the allowed number of votes to be casted per player per voting round

The judge then works on the copy of the transaction, that is not accessible to the player anymore

Here the judge indeed makes a copy of a copy so that the player cannot tamper with the copy either

www.websequencediagrams.com