Задание: построить формальную модель протокола консенсуса IBFT (Istanbul Byzantine Fault Tolerant) и проверить её методом Model Checking

Алгоритм IBFT вдохновлен алгоритмом PBFT и также решает проблему достижения консенсуса в распределенных системах, когда все корректные узлы должны принять решение о некотором общем предлагаемом значении.
Ссылка на оригинальную статью: https://arxiv.org/abs/2002.03613

Псевдокод алгоритма:

Algorithm 1 IBFT pseudocode for process p_i: constants, state variables, and ancillary procedures

```
1:  constants:
2:      p_i                                   ▷ The identifier of the process

3:  state variables:
4:      λ_i                          ▷ The identifier of the consensus instance
5:      r_i                                        ▷ The current round
6:      pr_i                  ▷ The round at which the process has prepared
7:      pv_i                  ▷ The value for which the process has prepared
8:      inputValue_i            ▷ The value passed as input to this instance

9:  timer:
10:     timer_i

11: procedure START(λ, value)
12:     λ_i ← λ
13:     r_i ← 1
14:     pr_i ← ⊥
15:     pv_i ← ⊥
16:     inputValue_i ← value
17:     if LEADER(h_i, r_i) = p_i then
18:         broadcast ⟨PRE-PREPARE, λ_i, r_i, inputValue_i⟩ message
19:     set timer_i to running and expire after t(r_i)
```

Algorithm 2 IBFT pseudocode for process p_i: normal case operation

```
1:  upon receiving a valid ⟨PRE-PREPARE, λ_i, r_i, value⟩ message m from LEADER(λ_i,
    round) such that JUSTIFYPREPREPARE(m) do
2:      set timer_i to running and expire after t(r_i)
3:      broadcast ⟨PREPARE, λ_i, r_i, value⟩

4:  upon receiving a quorum of valid ⟨PREPARE, λ_i, r_i, value⟩ messages do
5:      pr_i ← r_i
6:      pv_i ← value
7:      broadcast ⟨COMMIT, λ_i, r_i, value⟩

8:  upon receiving a quorum Q_commit of valid ⟨COMMIT, λ_i, round, value⟩ messages do
9:      set timer_i to stopped
10:     DECIDE(λ_i, value, Q_commit)
```

Algorithm 3 IBFT pseudocode for process p_i: round changes

1: **upon** $timer_i$ is `expired` **do**
2: $r_i \leftarrow r_i + 1$
3: set $timer_i$ to `running` and expire after $t(r_i)$
4: broadcast $\langle$`ROUND-CHANGE`, λ_i, r_i, pr_i, $pv_i\rangle$

5: **upon** receiving a set F_{rc} of $f+1$ valid $\langle$`ROUND-CHANGE`, λ_i, r_j, $\_$, $\_\rangle$ messages such that $\forall\langle$`ROUND-CHANGE`, λ_i, r_j, $\_$, $\_\rangle \in F_{rc} : r_j > r_i$ **do**
6: let $\langle$`ROUND-CHANGE`, h_i, r_{min}, $\_$, $\_\rangle \in F_{rc}$ such that:
7: $\forall\langle$`ROUND-CHANGE`, λ_i, r_j, $\_$, $\_\rangle \in F_{rc} : r_{min} \leq r_j$
8: $r_i \leftarrow r_{min}$
9: set $timer_i$ to `running` and expire after $t(r_i)$
10: broadcast $\langle$`ROUND-CHANGE`, λ_i, r_i, pr_i, $pv_i\rangle$

11: **upon** receiving a quorum Q_{rc} of valid $\langle$`ROUND-CHANGE`, λ_i, r_i, $\_$, $\_\rangle$ messages such that LEADER$(\lambda_i, r_i) = p_i \wedge$ JUSTIFYROUNDCHANGE(Q_{rc}) **do**
12: **if** HIGHESTPREPARED$(Q_{rc}) \neq \perp$ **then**
13: let v such that $(\_, v) =$ HIGHESTPREPARED$(Q_{rc}))$
14: **else**
15: let v such that $v = inputValue_i$
16: broadcast $\langle$`PRE-PREPARE`, λ_i, r_i, $v\rangle$

▷ We can omit this if we assume some mechanism external to the consensus algorithm that ensures synchronization of decided values.
17: **upon** receiving a valid $\langle$`ROUND-CHANGE`, λ_i, $\_$, $\_$, $\_\rangle$ message from $p_j \wedge p_i$ has decided by calling DECIDE$(\lambda_i, _, Q_{commit})$ **do**
18: send Q_{commit} to process p_j

Algorithm 4 IBFT pseudocode for process p_i: message justification

1: **predicate** JUSTIFYROUNDCHANGE(Q_{rc})
2: **return**
$\forall\langle$`ROUND-CHANGE`, λ_i, r_i, pr_j, $pv_j\rangle \in Q_{rc} : pr_j = \perp \wedge pv_j = \perp$
$\vee$ received a quorum of valid $\langle$`PREPARE`, λ_i, pr, $pv\rangle$ messages such that:
$(pr, pv) =$ HIGHESTPREPARED(Q_{rc})

3: **predicate** JUSTIFYPREPREPARE($\langle$`PRE-PREPARE`, λ_i, $round$, $value\rangle$)
4: **return**
$round = 1$
$\vee$ received a quorum Q_{rc} of valid $\langle$`ROUND-CHANGE`, λ_i, $round$, pr_j, $pv_j\rangle$ messages such that:
$\forall\langle$`ROUND-CHANGE`, λ_i, $round$, pr_j, $pv_j\rangle \in Q_{rc} : pr_j = \perp \wedge pr_j = \perp$
$\vee$ received a quorum of valid $\langle$`PREPARE`, λ_i, pr, $value\rangle$ messages such that:
$(pr, value) =$ HIGHESTPREPARED(Q_{rc})

▷ Helper function that returns a tuple (pr, pv) where pr and pv are, respectively, the prepared round and the prepared value of the `ROUND-CHANGE` message in Q_{rc} with the highest prepared round
5: **function** HIGHESTPREPARED(Q_{rc})
6: **return** (pr, pv) such that:
$\exists\langle$`ROUND-CHANGE`, λ_i, $round$, pr, $pv\rangle \in Q_{rc}$:
$\forall\langle$`ROUND-CHANGE`, λ_i, $round$, pr_j, $pv_j\rangle \in Q_{rc} : pr_j = \perp \vee pr \geq pr_j$

Задача заключается в построении формальной модели алгоритма (предпочтительно на TLA+) и дальнейшей проверке модели (model checking) в любой удобном инструменте (предпочтительно TLC).

Для модели алгоритма консенсуса необходимо проверить 3 основных свойства:

1. **Согласованность: все корректно работающие узлы принимают одинаковое значение.**

2. **Корректность: принятое значение было предложено одним из имеющихся узлов.**

3. **Завершаемость: все корректно работающие узлы достигают определённого значения.**