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

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

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

timer:
    timer_i

procedure START(λ, value)
    λ_i ← λ
    r_i ← 1
    pr_i ← ⊥
    pv_i ← ⊥
    inputValue_i ← value
    if LEADER(h_i, r_i) = p_i then
        broadcast ⟨PRE-PREPARE, λ_i, r_i, inputValue_i⟩ message
    set timer_i to running and expire after t(r_i)
```

Algorithm 2 IBFT pseudocode for process p_i: normal case operation

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

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

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

Algorithm 3 IBFT pseudocode for process p_i: round changes

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

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**
  let $\langle$`ROUND-CHANGE`, h_i, r_{min}, $_$, $_\rangle \in F_{rc}$ such that:
    $\forall\langle$`ROUND-CHANGE`, λ_i, r_j, $_$, $_\rangle \in F_{rc} : r_{min} \leq r_j$
  $r_i \leftarrow r_{min}$
  set $timer_i$ to `running` and expire after $t(r_i)$
  broadcast $\langle$`ROUND-CHANGE`, λ_i, r_i, pr_i, $pv_i\rangle$

upon receiving a quorum Q_{rc} of valid $\langle$`ROUND-CHANGE`, λ_i, r_i, $_$, $_\rangle$ messages such that $\textsc{Leader}(\lambda_i, r_i) = p_i \wedge \textsc{JustifyRoundChange}(Q_{rc})$ **do**
  if $\textsc{HighestPrepared}(Q_{rc}) \neq \perp$ **then**
    let v such that $(_, v) = \textsc{HighestPrepared}(Q_{rc}))$
  else
    let v such that $v = inputValue_i$
  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.

upon receiving a valid $\langle$`ROUND-CHANGE`, λ_i, $_$, $_$, $_\rangle$ message from $p_j \wedge p_i$ has decided by calling $\textsc{Decide}(\lambda_i, _, Q_{commit})$ **do**
  send Q_{commit} to process p_j

Algorithm 4 IBFT pseudocode for process p_i: message justification

predicate $\textsc{JustifyRoundChange}(Q_{rc})$
  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:
      $(pr, pv) = \textsc{HighestPrepared}(Q_{rc})$

predicate $\textsc{JustifyPrePrepare}(\langle$`PRE-PREPARE`, λ_i, $round$, $value\rangle)$
  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:
        $(pr, value) = \textsc{HighestPrepared}(Q_{rc})$

▷ 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

function $\textsc{HighestPrepared}(Q_{rc})$
  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.**